AFRL-RI-RS-TR-2015-021

COMPREHENSIVE ROUTING SECURITY DEVELOPMENT AND DEPLOYMENT FOR THE INTERNET

PARSONS GOVERNMENT SERVICES, INC.

FEBRUARY 2015

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ AIR FORCE MATERIEL COMMAND ■ UNITED STATES AIR FORCE ■ ROME, NY 13441

NOTICE AND SIGNATURE PAGE

FOR THE DIRECTOR:

/ S /
FRANK H. BORN
Work Unit Manager

/ S /
WARREN H. DEBANY, JR.
Technical Advisor, Information
Exploitation and Operations Division
Information Directorate

REPORT DOCUMENTATION PAGE			*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* February 2015	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* Sep 2009 – Oct 2014
4. TITLE AND SUBTITLE COMPREHENSIVE ROUTING SECURITY DEVELOPMENT AND DEPLOYMENT FOR THE INTERNET		**5a. CONTRACT NUMBER** FA8750-09-C-0192
		5b. GRANT NUMBER N/A
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Sandra Murphy		**5d. PROJECT NUMBER** DHSB
		5e. TASK NUMBER GP
		5f. WORK UNIT NUMBER 09
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Parsons Government Services, Inc. 25531 Commercenter Dr STE 120 Lake Forest CA 92630-8874		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIGA 525 Brooks Road Rome NY 13441-4505		**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
		11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2015-021
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. PA# 88ABW-2015-0247 Date Cleared: 22 Jan 2015		
13. SUPPLEMENTARY NOTES		

14. ABSTRACT

The objective of this effort is to design, implement and facilitate deployment of a comprehensive solution for security for Border Gateway Protocol (BGP), stardardizing the solution in the Internet Engineering Task Force (IETF). The project resulted in published IETF documents that are in production use in all regions of the Internet. The IETF specifications provide a Resource Public Key Infrastructure (RPKI) for strong origin validation protection for BGP routes. The project produced implementations and tools to facilitate deployment and monitor use. The project team assembled a design team to design and specify a path validation solution, creating a comprehensive solution. This path validation design was submitted to the IETF where it is a work in progress. The project team implemented the path validation specification.

15. SUBJECT TERMS

Internet Routing, Infrastructure, Border Gateway Protocol, BGP, Resource Public Key Infrastructure, RPKI

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** FRANK BORN
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	58	**19b. TELEPHONE NUMBER** *(Include area code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

List of Figures and Tables

List of Figures

Acknowledgements

The following are acknowledged for their contributions to this report and to the work of this project:

Rob Austein

Michael Baer

Edric Barnes

Randy Bush

Andrew Chi

Michael Elkins

Wes Hardaker

Stephen Kent

Derrick Kong

Matthew Lepinski

David Mandelberg

Russ Mundy

Mark Reynolds

Karen Seo

Anuja Sonalker

Robert Story

Ronald Watro

Samuel Weiler

1 SUMMARY

The Internet is a critical resource both nationally and internationally. It must provide stable, reliable operation for the millions of people, organizations and governments that depend upon it. The world's critical communication, emergency, energy, control and financial systems are irrevocably tied to the Internet.

The Internet routing infrastructure provides the basic capability for Internet traffic to be exchanged between millions of end users. Forwarding traffic between any users of the global Internet is provided by routing infrastructure that exchanges routing information. There is only one protocol used in the global Internet for this exchange, the Border Gateway Protocol (BGP). However, BGP is recognized as being vulnerable to misuse and attack that could damage the exchange of traffic in the Internet.

The objective of this effort is to design, implement and facilitate deployment of a comprehensive solution for security for BGP, encompassing both validation of the origin of a BGP announcement and validation of the path it takes. To ensure a comprehensive solution, work in this project has two parallel goals: to bring to fruition the standardization and deployment of origin validation, and to design, standardize, and implement a compatible path validation solution.

This project has had great success. As the project began, there were works in progress in the Internet Engineering Task Force (IETF) to define a security architecture for BGP origin validation, but no published standards and no certification services in production At this point:

- The IETF has published a set of documents that provide a security architecture for BGP origin validation, called the Resource Public Key Infrastructure (RPKI). Of the two dozen published documents, eighteen are co-authored by members of this team.
- Every Regional Internet Registry (RIR) is certifying resources under this security architecture.
- Major router vendors have implemented the use of the RPKI.
- Workshops, tutorials and presentations have spread the word in operational comunities worldwide.
- A path validation solution, BGPSEC, has been designed, specified, and accepted by the IETF working group as a work item.
- First announcements of planned or initiated use of RPKI data in routing operations have made.
- Government agencies have issued recommendations for certification in general or RPKI use in particular.
- Major network service providers have begun testing and experimentation in labs or off-line

This progress is astounding after decades of BGP security proposals that did not succeed. However, there remain challenges:

- Like all new technologies, deployment of the RPKI requires cost and effort. And, in common with all security technologies, deployment does not show an immediate, direct, clear return on investment for first and early adopters. Benefit grows as the number of adopters grow.

- The RPKI parallels the address allocation hierarchy, and some fear mis-behavior by a hierarchical authority (whether forced, deliberate, or accidental) could do damage to routing.
- The transport protocol for the RPKI is recognized to have scaling problems.

2 INTRODUCTION

The Internet is a critical resource both nationally and internationally. It must provide stable, reliable operation for the millions of people, organizations and governments that depend upon it. The world's critical communication, emergency, energy, control and financial systems are irrevocably tied to the Internet.

The Internet routing infrastructure provides the basic capability for Internet traffic to be exchanged between millions of end users. Forwarding traffic between users of the global Internet is provided by routing infrastructure that exchanges routing information. There is only one protocol used in the global Internet for this exchange, the BGP.

The President's National Strategy to Secure Cyberspace [35] recognizes the need for essential improvements to the routing system of the Internet. Because of its ubiquitous use, exploitation of BGP vulnerabilities has global reach and impact on Internet traffic. The National Strategy specifically states:

> The Border Gateway Protocol (BGP) is at greatest risk of being the target of attacks designed to disrupt or degrade service on a large scale.

The National Academy of Sciences's book *At the Nexus of Cybersecurity and Public Policy: Some Basic Concepts and Issues* [34] recently echoed this concern:

> ... the Internet is a network of networks. Each network acts as an autonomous system under a common administration and with common routing policies. BGP is the Internet protocol used to characterize every network to each other, and in particular to every network operated by an Internet service provider (ISP).
>
> In general, the characterization is provided by the ISP responsible for the network, and in part the characterization specifies how that ISP would route traffic to a given destination. A problem arises if and when a malicious ISP in some part of the Internet falsely asserts that it is the right path to a given destination (i.e., it asserts that it would forward traffic to a destination but in fact would not). Traffic sent to that destination can be discarded, causing that destination to appear to be off the net. Further, the malicious ISP might be able to mimic the expected behavior of the correct destination, fooling unsuspecting users into thinking that their traffic has been delivered properly and thus causing further damage.

An illustration of the potential for damage is the YouTube incident [43]. When officials in Pakistan wanted to block content from www.youtube.com from reaching their country, BGP mechanisms were used to stop the content from reaching Pakistan. However, these BGP mechanisms were not restricted to Pakistan and also prevented Internet users outside Pakistan from reaching www.youtube.com. In the end, every Internet user was impacted, not just those within the nation state of Pakistan. There was nothing especially vulnerable about YouTube; the same could happen to any site in the Internet.

For over a decade prior to this project, many different security protections for BGP had been proposed but had failed to be deployed. The various communities critical for deployment (network operators, resource registries, router vendors, standardization communities, security communities) did not accept, implement and deploy any of these approaches.

2.1 BGP and a Comprehensive Solution

BGP provides propagation of network *reachability information* to Internet addresses among the Autonomous Systems (ASs) that make up the global Internet. An AS uses the reachability information it has received in order to forward packets to their destination.

The BGP protocol produces a route to a network address in two steps. First, an AS (e.g., an enterprise, an Internet Service Provider (ISP) network, etc.) that has direct connectivity to a network that is using that address originates a BGP route for that address. Second, an autonomous system propagates BGP routes it has received from one neighbor to other neighbors, adding its Autonomous System Number (ASN) to the path of ASs recorded in the BGP route.

There are two different attacks that can be conducted on these two steps. First, an AS may mis-originate a BGP route, falsely claiming direct connectivity to an address. Second, an AS may path spoof a BGP route, inventing or corrupting the path of ASs recorded in the route. Both steps may result in traffic being misdirected away from the legitimate destination or away from the intended route.

The IETF, with leadership and participation from members of this team, began work on a security architecture that would protect BGP originations. However, protection of BGP originations (*origin validation*), even when brought to full deployment and use, is only the initial, base step of BGP security. A BGP route must also include a legitimate path, verified through *path validation*, to the authorized origin. Otherwise, an authorized origin can be grafted onto a bogus BGP route and the bogus advertisement will still be accepted.

For comprehensive security in the Internet routing infrastructure, both origin validation and path validation must be assured.

The objective of this effort is to design, implement and facilitate deployment of a comprehensive solution for security for the Border Gateway Protocol (BGP), completing the security basis underway in the Internet Engineering Task Force (IETF). To ensure a comprehensive solution, work in this project had two parallel goals: to bring to fruition the standardization and deployment of origin validation, building on the work in progress in the IETF, and to design, standardize, and implement a compatible path validation solution. Results of this work will be made freely and openly available without restriction, in order to facilitate the widest possible use in the Internet including use by commercial, research and academic organizations. The remainder of this document focuses on the work that was done, and the progress made toward these goals.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Project Design

We designed this project around several assumptions: that the project team constituency would be critical to the success of an acceptable, deployable, and secure solution, that implementation and particularly open source implementation would be needed to ensure success, that a solution would need to mirror the existing routing infrastructure processes and structures, that a comprehensive solution strategy needed distinct independent strategies for the two solution components, and that proactive assistance for deployment would be needed.

3.1.1 Project Team: Multi-Stakeholder Constituency. The success of this project required the acceptance and deployment by several key stakeholder communities.

To ensure that success, we created a project team that included experienced representatives from many of the critical communities - those familiar with network infrastructures operations, with open standardization, with BGP, with the registry community, with software implementation for network applications, and with security -

The PARSONS team included

- Parsons: Parsons brought decades of experience in security, particularly routing security, in the IETF standardization process including leadership roles, and in the design and deployment of secure Internet infrastructure protocols.
- ISC: ISC brought decades of experience, including leadership roles, in software development, in distributed infrastructure protocols, in the IETF standardization process, in open source development, and in the software platform that formed the basis for this work.
- RGNET: RGNET brought decades of experience, including leadership roles, in network operations and network operations communities, in Internet registries and their communities, in the IETF standardization process, and in router vendor software development.
- DRL: Dragon Research Labs is composed of the critical key personnel of ISC and RGNET.
- BBN: Raytheon BBN Technologies brought decades of experience, including leadership roles, in security, in routing security, and in production level software development.

3.1.2 Implementation: Reference Implementations and Open Source Model. This project's objective requires acceptance, deployment and use of a new infrastructure protocol. That objective required standardization for wide implementation by commercial vendors, and testing and experimentation on a global scale. Standardization produces useful results best if there is simultaneous prototyping of the works in progress, to feed back implementation experience into the standardization work. Implementation experience is also a requirement for progress in the IETF standardization process. Reference implementations also serve others who are implementing or testing implementations.

The PARSONS team realized that freely and openly available software implementations were a critical method in this project. Open source reference implementations

- provide a vehicle for academic study

- provide a vehicle for experimentation and study by users (registries, network operators, etc) prior to deployment
- avoid the barriers to early adoption and use that come from restrictions on use
- provide a reference and basis for future commercial development
- provide an implementation for network operators to use internally operationally before or instead of commercial implementations

The software methods we used are described in Section 3.2.

3.1.3 Security Solutions: Parallel Existing Systems. Security solutions function best (or at all) when they parallel closely the systems being protected. The close parallel ensures that the structures and behaviors of the system can be matched by structures and behaviors of the security solution. We followed that paradigm in designing the comprehensive BGP security solution, paralleling the BGP steps in producing a route and paralleling the existing system for allocating address space.

As mentioned, the BGP protocol produces a route to a network address in two steps. First, an autonomous system (AS) (e.g., an enterprise, an Internet Service Provider network, etc.) originates a BGP route. This origin autonomous system creates a BGP route including only its autonomous system number (ASN). Second, an autonomous system propagates routes it has received from one neighbor to other neighbors, appending its ASN to the path of ASs recorded in the BGP route.

The comprehensive BGP security solution parallels the two steps of producing a BGP route to an address.

The initial, base step of the comprehensive BGP security solution parallels the initial, base step of producing a BGP route - by providing assurance that the AS originating a route has the authority to do so. Only one with the right to use an address may grant the authority to originate a route to the address. An important component of this part of the solution is providing assurance of that right to use an address.

The right to use an address derives from the Internet resource allocation system. In this system, Internet Assigned Numbers Authority (IANA) holds the authority to allocate Internet network resources, including addresses and ASNs, for use in the Internet. IANA allocates large blocks of network addresses to the Regional Internet Registries (RIRs) that are responsible for address allocations to their members in their respective regions. (A block of address space is sometimes called an address *prefix*.) The RIR members may further suballocate from the addresses they receive to their members or customers. In any allocation, a resource holder may suballocate only from the address space it itself holds. The prefix allocation system provides a tree of the address allocations and suballocations.

The IETF work in progress at the start of this project had begun to define a RPKI architecture that provides strong cryptographic assurance of the right to use an address. Because the RPKI is designed to protect the address allocation system, it was designed to mirror the allocation system, so that each allocation is mirrored by a cryptographic assertion of the allocated address space. The RPKI structure parallels the tree structure of the address allocation system.

Based on the RPKI assurance of the right to use the address. the IETF work in progress provided cryptographic assurance that an AS is authorized to originate a BGP route to a address (a Route Origin Authorization (ROA)).

The second step of the comprehensive BGP security solution parallels the second step of BGP route creation, by providing assurance that the path recorded in the BGP route was propagated properly by each AS mentioned in the path. The origin validation assurance is the proper base step for the solution, since the base step for construction of a BGP route is origination. However, this is not a comprehensive solution. With origin validation alone, an AS might construct a path for a BGP route with a valid origin and propagate the route as if it had received it from the authorized origin. That would circumvent the origin validation protections, and would produce a corrupt path.

In our design of a comprehensive solution, we took care that the path validation protection mirrored the BGP route propagation step. The protections are carried in a BGP attribute that is augmented when the AS path in the BGP route is augmented. The attribute demonstrates that the AS path was constructed properly as the BGP route was propagated. Special considerations were introduced to parallel recognized common legitimate BGP operations.

3.1.4 Comprehensive Solution: Distinct Strategies for Solution Components. Because origin validation can be established by reference to information external to the BGP protocol (the right to use an address), but path validation relies on the behavior of the BGP protocol (the propagation of a BGP route in a BGP Update), the strategies of the two solution components were quite different.

Origin Validation Strategy. At the time this project started, the IETF had already adopted the RPKI and origin validation as standardization activity in the IETF Secure Inter-Domain Routing (SIDR) working group. However, the specification documents were still works in progress and there had been no deployment.

The goal of the origin validation component of this work was to advance the documents in the IETF standards process to publication, to simultaneously and continuously maintain a reference implementation, and to work with the community to begin test, evaluation, production and deployment.

As an outgrowth of this methodology, the PARSONS team introduced new features to the architecture in recognition of deployment barriers. Of particular note are local RPKI caches and local control over trust decisions. The PARSONS team introduced local RPKI caches in recognition that it would be burdensome for a router to maintain synchronization with the global RPKI system and to perform the RPKI cryptographic checks necessary. RPKI caches local to a network perform the global synchronization and validation and communicate only the information necessary for origin validation to the router. The PARSONS team introduced local trust anchor management [30] as a means by which an AS could impose constraints on the RPKI data and give priority to local trust decisions. This allowed each AS to have distinct private RFC1918 [42] address space, or override RPKI data for address space it considers its own, for example.

Path Validation Strategy. As this project began, the IETF had not yet adopted any standardization activity to address BGP path validation. There was no identified solution for path validation. Consequently, the methods chosen to produce this component of a comprehensive solution were much different from origin validation.

The PARSONS team chose to assemble an ad-hoc volunteer design team, supplementing the PARSONS team with additional personnel from the operations, security and policy communities. The design team membership was fluid, as others with insight into particular topics were asked to participate from time to time. This design team interacted regularly to design and polish a path validation solution.

When the path validation solution was deemed to meet the IETF expectations for an adequate start for standardization activity, the team submitted the design to the SIDR working group for adoption. From that point on, the PARSONS team continued to pursue standardization within the IETF process, where the path validation solution was named BGPSEC.

As support of the standardization process, the PARSONS team produced a reference implementation of the path validation solution.

3.1.5 Deployment: Proactive Advocacy. In order to further test and acceptance in the community, the PARSONS team, particularly DRL, began to provide tutorials to the important stakeholder communities. From this grew hands-on workshops, where participants could use the RPKI reference implementations in real time. Feedback from the participants, and recognition of their desire to see origin validation work in an operational environment, led to changes to the workshop platform. Eventually, the workshop provided hands-on, real-time experience with all steps of the use of the RPKI, from initial certification of resources and creating their own CA to configuration of an Internet connected router running commercial router vendor releases and experimenting with RPKI based routing decision policies. The PARSONS team recognized the operators need to experience the use of this new technology in familiar operational settings.

The PARSONS team chose a paradigm of frequent interaction with other RPKI implementers and users, doing interoperability testing at common venues and assembling implementers and users for “hack-a-thons”, to jointly build and test software in a communal environment. This paradigm proved beneficial as the communal environment provided quick resolution of individual problems and strong immediate feedback of issues to the respective implementers.

As policy issues began to arise outside and inside the technical communities, the PARSONS team advocated for this solution. Where concerns arose from uncertainty or misunderstanding, the PARSONS team provided accurate information about the features of the technology.

3.2 Software Reference Implementation Designs

This project produced three different software packages:

rpki.net: *rpki.net* is an open source implementation of both the Certification Authority (CA) and the Relying Party (RP) functions in the RPKI. As part of the relying party function, it includes an implementation of the rpki-rtr protocol that provides communication of RPKI data between an RPKI cache and a router.

RPSTIR: *Relying Party Security Technology for Internet Routing (RPSTIR)* is an open source implementation of the relying party functions in the RPKI, including an implementation of the rpki-rtr protocol [10] and local trust anchor management [30], as well as a stringent set of tests to prove a relying party implementation is correctly detecting errors in RPKI certificates. A relying party implementation passing these stringent tests can be used to rigorously test a CA's compliance with the RPKI specifications.

BGPSEC: *BGP Security (BGPSEC)* is an open source implementation of the BGPSEC path validation protocol specification that is a work in progress [32] in the IETF SIDR working group.

Together these implementations provide a complete environment for a comprehensive solution for BGP security in the internet, both origin validation and path validation. The three software packages are discussed separately in the following sections.

3.2.1 Software: Rpki.net. The rpki.net software package implements both the production (CA) and relying party (RP) functions of an RPKI environment.

Figure 1, below, shows the overview of the rpki.net architecture. See the rpki.net document "RPKI Tools Manual" [45] for more information.

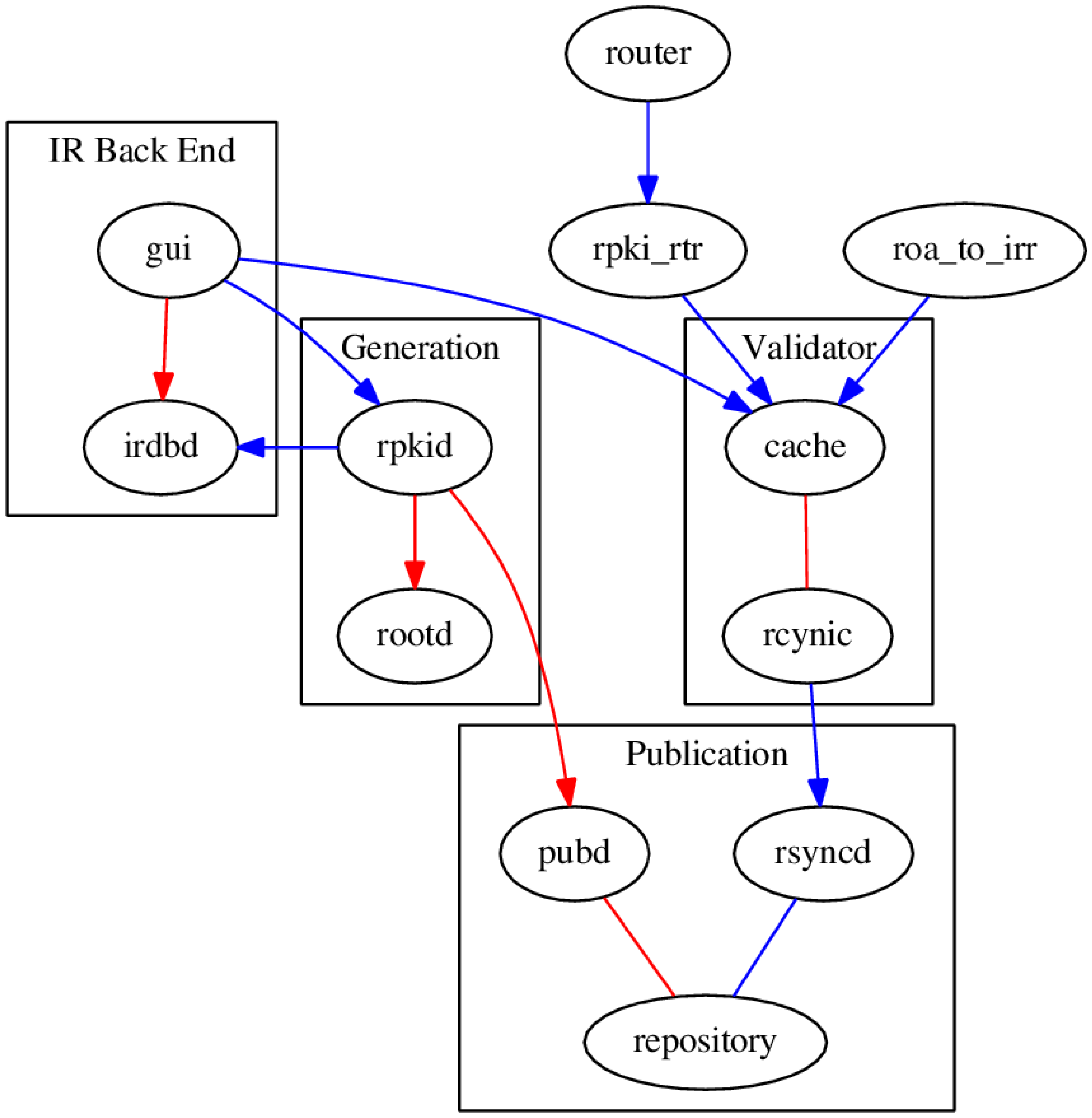

Figure 1: rpki.net

3.2.1.1 Relying Party Tools. Those who operate routers and want to use RPKI data to help secure them, will make use of the relying party tools. These tools implement the "relying party" role of the RPKI system, that is, the entity which retrieves RPKI objects from repositories, validates them, and uses the result of that validation process as input to other processes, such as BGP security.

The RP main tools are rcynic and rtr-origin. rcynic is the primary validation tool. It does the actual work of RPKI validation: checking syntax, signatures, expiration times, and conformance to the profiles for RPKI objects. The other relying party programs take rcynic’s output as their input. rtr-origin is an implementation of the rpki-rtr protocol, using rcynic’s output as its data source. rtr-origin includes the rpki-rtr server, a test client, and a utility for examining the content of the database rtr-origin generates from the data supplied by rcynic.

3.2.1.2 Certificate Authority (CA) Tools. Those who control RPKI resources and need an engine that supports requesting certificates, issuing ROAs, or issuing certificates to other entities, will make use of the CA tools.

The RPKI CA engine is an implementation of the production-side tools for generating certificates, CRLs, ROAs, and other RPKI objects. The CA tools are implemented primarily in Python, with an extension module linked against an RFC-3779-enabled [29] version of the OpenSSL libraries to handle some of the low-level X.509 details.

(Since CAs are generally also relying parties (if only so that they can check the results of their own actions), network operators who operate a CA will likely use the relying party tools as well.)

The RPKI CA engine consists of a certificate issuance engine, a publication engine, and a registry backend. The separation of the certificate issuance engine from the publication engine allows for multiple certificates issuance engines to use the same publication engine. This also allows the publication to be publicly accessible, as is necessary, but the certificate issuance engine to be more protected.

The certificate issuance engine has a protocol (officially called the provisioning protocol [18], but known commonly as the up-down protocol) for making requests to its parent in the RPKI tree and responding to requests from its RPKI children. The publication engine has a publication protocol to speak to certificate issuance engines [48]. The certificate issuance engine and the registry backend communicate with a protocol called the “left-right” protocol. (Because it is expected that the CA backend processes and databases will be a local choice, there was no attempts to standardize the left-right protocol.)

The RPKI CA engine includes the following programs:

- **rpkid:** rpkid is the main RPKI certificate issuance engine daemon.
- **pubd:** pubd is the publication engine daemon.
- **rootd:]** rootd is a separate daemon for handling the root of an RPKI certificate tree. This is essentially a stripped down version of rpkid with no database, no left-right protocol implementation, and only the parent side of the up-down protocol. It’s separate because the root is a special case in several ways and it was simpler to keep the special cases out of the main daemon.

- **IRBE:** irdbd, rpkic, and the GUI collectively make up the "Internet registry back end" (IRBE) component of the system.
  - **irdbd:** irdbd is a sample implementation of an Internet Registry Database (IRDB) daemon. rpkid calls into this to perform lookups via the left-right protocol.
  - **rpkic:** rpkic is a command line interface to control rpkid and pubd.
  - **GUI:** A web-based graphical interface to control rpkid and pubd(described in section 3.2.1.3).

3.2.1.3 Rpki.net GUI Interface. In order to make the system easier to operate for end users, the need for a graphical interface to serve as a front end to the RPKI became apparent. A HTTP-based system was deemed the best way to achieve cross platform availability since every end user system has a graphical web browser available.

On the server side, the choice was made to make use of the Django [37] web framework, which allows for rapid prototyping of web applications. Django is open source, widely used, has excellent documentation, and has a very active community to draw on for help.

The initial design goal for the web interface was to provide a "dashboard" that a user could view to get an overview of all RPKI resources under the user’s control. This dashboard presents information about: RPKI parents, children, repositories, ROAs. The dashboard view also attempts to draw the user’s attention to resources that are not covered by any ROA, in order to prevent inadvertent "unknown" validation results.

The web interface also helps perform some validation when the user attempts to create a ROA, such as ensuring that the resources are actually under the user’s control.

Internally, the web interface is a front end to the IRDB and rpkid daemons. The web interface makes queries to rpkid to fetch the list of RPKI resources under a particular user’s control, and this information is stored in a database, from which the display in the web application is generated. As the user requests changes, such as creation of ROAs, the web interface makes changes to the IRDB, and notifies rpkid that it should contact irdbd to process those changes.

Since a resource holder is also interested in how other relying parties may view their origin authorizations, the web interface also allows the user to see the validity status of all routes that are covered by resources listed in their resource certificates. In order to perform this step, two additional sources of information need to used: the global RPKI for route authorizations, and RouteViews [44] for a snapshot of the current globally announced routes.

The snapshot of the global RPKI is generated by processing the output of the rcynic tool, described in section 3.2.1.1. This output consists of an Extensible Markup Language (XML) file, and a directory of resource certificates, ROAs, and Ghostbuster objects. For each object that was considered valid within the RPKI, the web interface software reads the repository objects and extracts information into a database that drives the web application display. Using this data, it is possible to perform the RPKI route origin validity checks that other relying parties will conduct, and give visual feedback to the user if there are route origins that are invalid or unknown.

The RouteViews project produces a dump of the global routing table every two hours. This dump is a merge of views from multiple sensors, so it gives a very good overview of the actual state of the global routing tables. A batch job runs in the background every two hours to fetch this data and import it into the database used by the web application. This allows the web interface to quickly find globally announced routes that are covered by RPKI resource certs and ROAs. Combined with the data from the global RPKI, the validity status of each route can be calculated and displayed for the user.

In the next design phase, a need for alerting the user about potential problems with their route origin validity outside of the web interface was identified. For example, if a resource certificate somewhere in the chain will expire soon, the user may want to be notified immediately via an email rather than waiting until the next time the user opens the web interface. Since data from the global RPKI is updated hourly, and the RouteViews data is updated every two hours, feedback to the user can be generated relatively quickly in order to rectify any problems. In addition to the expiration checks mentioned above, other types of common problems that can be detected are: changes in resource delegated to a resource holder, route origin validation status changes from valid to unknown or invalid, and new routes appearing that are not covered by existing ROAs. Where there are Ghostbuster objects published for resource certificates up the chain, the alert can provide human contact information from those records. This can be useful in the situation where the contact for a resource holder’s RPKI parent can’t immediately be contacted, and it may not be obvious how to contact the RPKI grandparent.

3.2.1.4 Documentation and Development Environment. The rpki.net software is kept in a subversion repository, from which binary packages are built nightly for a small set of popular unix operating systems.

Documentation for the tools is written on the wiki hosted at rpki.net. The wiki pages are organized in a way that allows automatic conversion to a full document covering installation and use, for offline reading.

3.2.1.5 Software Dependencies. In keeping with the decision that the rpki.net software should be freely and openly available, the rpki.net software makes extensive use of several open software packages and libraries. This also allowed the implementers to focus on the features that are RPKI unique. The packages were chosen for their wide use, active user community for support and active implementation community for feature enhancement and bug fixes.

- **MySQL:** MySQL is a widely used and popular open source database package. It was chosen for database support in the rpki.net package.
- **Apache:** Apache is a commercial grade web server package used in the rpki.net user interface.
- **OpenSSL:** OpenSSL is a widely used implementation of many cryptographic features, including support for X.509 Public Key Infrastructure, XML, and many different cryptographic algorithms.
- **Python:** Python is a general purpose, high level programming language, recognized as being easy to learn, providing compact and efficiency in coding, and as a consequence providing ease of implementation and lower maintenance burden.
- **libxml:** libxml is a library providing an API for manipulating Extensible Markup Language (XML) data

- **YaML:** YaML is a human-readable data serialization format for all programming languages
- **Django:** Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

3.2.2 Software: RPSTIR. The PARSONS team member Raytheon BBN Technologies created an open-source release of the Relying Party Security Technology for Internet Routing (RPSTIR, pronounced "rip-stir"). RPSTIR provides relying party tools to retrieve and verify Resource Public Key Infrastructure (RPKI) objects contained in the world-wide system of RPKI repositories.

RPSTIR helps network operators detect and reject accidental, false route origin advertisements, thus reducing the likelihood of inadvertent Internet address space hijacking.

RPSTIR synchronizes with the global RPKI repository system, verifies the data, and extracts a list of authorized prefix-origin AS pairs. This list is precisely the information a router requires in order to detect false BGP origin announcements.

RPSTIR implements the RPKI to Router Protocol (rpki-rtr, RFC6810[10]), allowing routers to communicate with RPSTIR to verify route origin announcements and detect false origin announcements due to errors by network operators (e.g., the Pakistan Telecom hijack of YouTube address space [43]).

RPSTIR also implements the local trust anchor management [30].

Figure 2, below, shows the overview of the RPSTIR architecture.

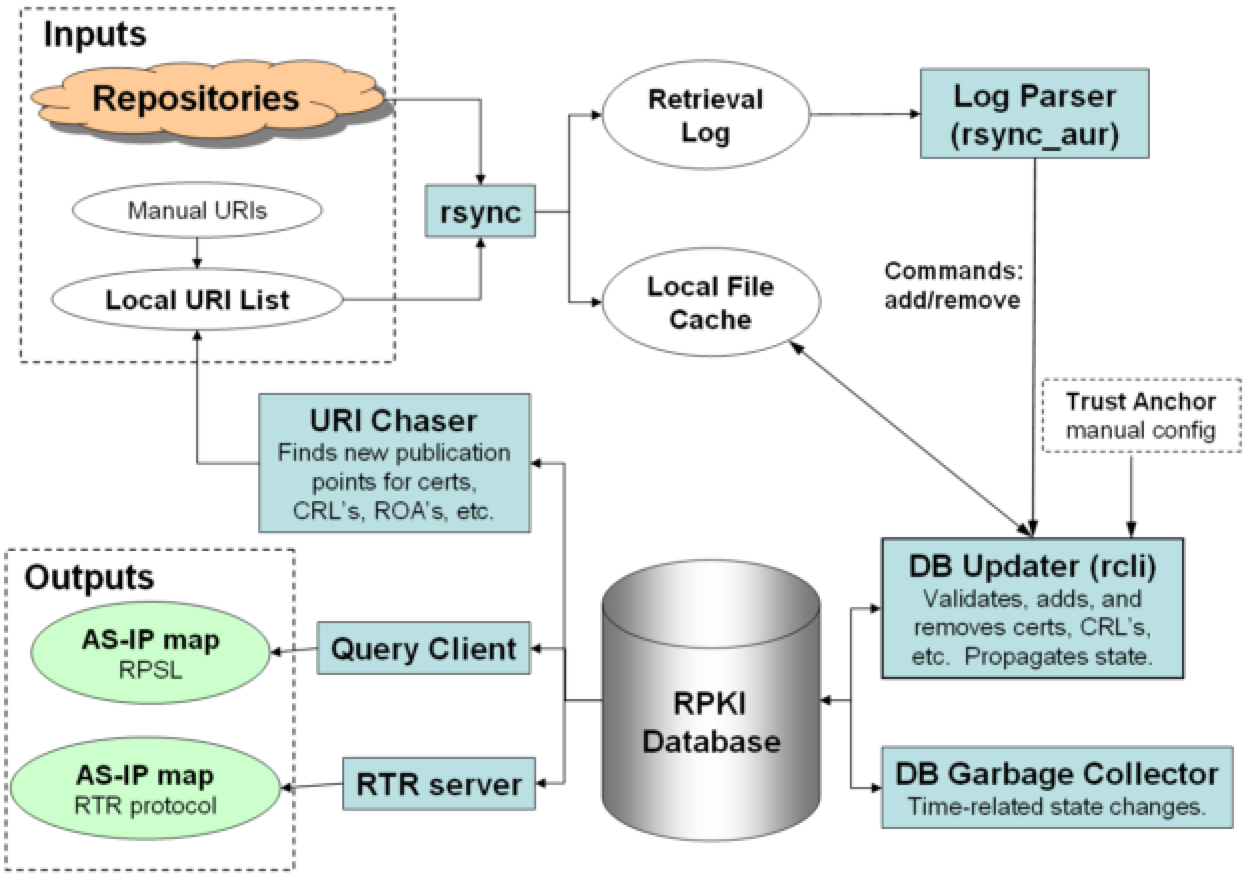

Figure 2: Overview of RPSTIR Implementation

The RPSTIR software is composed of several components.

3.2.2.1 RPKI synchronization and local caching. The *rpstir-synchronize* utility of RPSTIR uses the rsync protocol to synchronize with the global RPKI repository system and create a local rsync file cache. The utility must be run periodically to keep it in sync with the global RPKI. Users must configure their system using Cron or another time based job scheduler to run *rpstir-synchronize* at an interval appropriate for their system.

After the local file cache has been synchronized, the *rsync_aur* utilty parses the sychronization log file and passes updated configuration information to the *rcli* utility. The *rcli* utility validates the RPKI objects and extracts a list of authorized prefix-origin AS pairs to a local RPKI database cache. This local database cache can be queried using the *rpstir-query* utility.

3.2.2.2 RTR Server; Route Origin Verification. RPSTIR implements the RPKI to Router Protocol (rpki-rtr, RTR), which creates flexibility in the deployment of RPKI in a network. Using the rpki-rtr protocol, RPSTIR can be hosted anywhere in the router's network, sparing the router the burden of the synchronization and cryptographic validation. RPSTIR would produce the local RPKI database cache and act as an rpki-rtr server. Routers could communicate with the RPSTIR rpki-rtr server to retrieve the list of authorized prefix-origin AS pairs, and use that information to verify route origin announcementa and detect false origin announcements.

The *rpstir-rpki-rtr-update* utility updates the local rpki-rtr cache and should be called after *rpstir-synchronize*.

RPSTIR also offers an Routing Policy Specification Language (RPSL) output option, enabling operators to generate route filters compatible with existing, deployed router and operations software.

3.2.2.3 Compliance Tools. RPSTIR provides fine-grained, stringent compliance tests for relying party code. These test cases can be used to test any relying party implementations for compliance with published RFCs and Internet-Drafts, testing that compliant RPKI objects pass the tests and non-compliant objects fail. Additionally, relying party software that passes the tests can be used to test the output of a CA, to ensure that the CA is producing compliant products.

3.2.2.4 Software Dependencies. RPSTIR depends on several other open source packages.

- **MySQL:** MySQL is used for the the local RPKI database cache.
- **OpenSSL:** OpenSSL is used for cryptographic libraries for X.509 certificates.
- **ODBC mySql Connector:** ODBC (Open Database Connectivity) is a standard programming interface (API) for accessing database, used to connect with the local RPKI database cache.
- **rsync:** Rsync is used to synchronize a local file cache of global RPKI data.
- **Python:** Python is a programming language that lets you work quickly and integrate systems more effectively.
- **Netcat:** Netcat is a networking utility which reads and writes data across network connections.
- **Cryptlib:** Cryptlib is a library which provides implementations of complete security services via a simple high level programming interface (API).
- **cURL:** cURL is a command line tool and library for transferring data with Universal Resource Locator (URL) syntax (e.g. HTTP, FTP, etc).
- **OpenSSH:** OpenSSH provides utilities and libraries for Secure Shell (SSH) access.

3.2.3 Software: BGPSEC Implementation. As part of developing and deploying BGPSEC, this project produced a proof-of-concept implementation of the BGPSEC protocol [32]. The main goals for the implementation are:

- An open source, freely available reference implementation of the BGPSEC protocol.
    - For academic study
    - For ISP internal use/testing

- A functional software router to provide operational experience and feedback into the protocol design.
- Proof-of-concept code suitable to support advancement of the BGPSEC specifications within the IETF.
- A basis for future development and use of the protocol.

In order to fulfill these goals, a survey of open source routing software was conducted to determine both whether we should create our own software package or build off of one the existing software packages and which one to use if we chose to go with existing router software. The choice of the BIRD [7] software package determined the programming language used: C.

BIRD provides an implementation of BGP that includes origin validation. The BIRD origin validation relies on an external ROA table that lists the authorized AS originators of a route to any prefix. A process external to BIRD is expected to populate that table, in order that the origin validation not be tied to any particular method of supplying that information.

The RPKI is a source of the authorized AS originators information that is needed. It also provides the public router keys. Our BGPSEC implementation provides the required external process, bgpsec-bird-client, which retrieves the authorized AS and prefix information from an RPKI cache and populates BIRD's internal ROA table. It also retrieves router public keys to make them available for the BGPSEC code.

The build process was also determined by the choice of routing software: automake. The cryptographic code within the BGPSEC implementation also requires OpenSSL libraries that support Elliptic Curve Digital Signature Algorithm (ECDSA).. The bgpsec-bird-client requires a version of RTRlib that supports router key retrieval. A multi-user accessible source code version control system was created using git for code development. The implementation developers interacted with he Secure Inter-Domain Routing (SIDR) Working Group [49] and the protocol developers to provide feedback for the protocol development.

4 RESULTS AND DISCUSSION

The PARSONS team’s efforts have directly provided or supported a great leap forward in the security of the Internet’s routing infrastructure.

4.1 Comprehensive Solution

The comprehensive solution developed in this project encompasses both origin validation and path validation.

Origin validation is based on the right to use a prefix. That right to use is represented and demonstrated in the RPKI developed in the IETF SIDR working group. The RPKI [26] mirrors the address allocation system. An address allocation is represented as an X.509 certificate [21] that binds a public key to a list of resources, i.e., addresses or ASNs. Each ISP allocated an address can be issued a CA certificate for the allocation and can then issue certificates when it suballocates to customers. The RPKI includes the ability to reclaim an address by revoking the certificate, represented in a CRL.

The authorization to originate a BGP route to an address is represented in a Route Origin Authorization (ROA) [27], a signed object that is verified by an RPKI certificate for the address. If and when the authorization is no longer appropriate, the ROA can be revoked by revoking the certificate that issued it.

Each CA maintains a repository, or publication point, [19] of all the certificates and ROAs it has produced. The repository also contains a manifest of all the contents [3] and a ghostbuster record [8] giving contact information.

The relying parties retrieve the RPKI certificates and signed objects from the globally distributed system of RPKI repositories and cryptographically validate the certificates and objects. The pairs of authorized AS and prefix can then be used to judge the validity of a BGP route [33].

The deployment architecture includes a local RPKI cache, which is responsible for synchronization with the global system of RPKI repositories and validation of what it retrieves. The local RPKI cache communicates just the authorized AS - prefix pairs to a router in the RPKI to Router protocol, known as rpki-rtr [10]. The rpki-rtr protocol is a light-weight query/response protocol that allows a router to retrieve just the extracts of RPKI data needed to validate BGP routes from a local RPKI cache.

Local trust anchor management [30] introduced a means by which an AS could impose constraints on the RPKI data in accordance with local trust decisions. This allowed each AS to have distinct private RFC1918 [42] address space, or override RPKI data for address space it considers its own, for example.

Considerations of operational use are described in [9], with consideration of key or algorithm change in [20] and [16].

The path validation component provides cryptographic assurance that a BGP route was propagated through the ASs that appear in the AS path carried in the route. This prevents path spoofing and also prevents circumvention of the origin validation protections. The protocol extension to BGP that provides the path validation is called BGPSEC [32]. BGPSEC defines a new attribute to be carried in a BGP route. A BGPSEC capable router adds a new attribute segment to the BGP routes it sends. Each BGPSEC attribute segment contains a signature of the received route's signature (which protects the received AS path), the origin AS, the prefix, and the AS to which the BGP route is sent. Including the recipient AS in the signature prevents cutting and pasting sections of protected paths in order to produce a spoofed path. The recipients check the signatures on the attributes to establish that the path valid.

Because each router produces signatures for BGPSEC attributes on the BGP routes it sends, it needs a key to use to sign the attributes and a certificate others can use to verify that signature. A work in progress in the SIDR working group [31]specifies a new certificate in the RPKI to certify a router's public key, which must be issued by the AS to which the router belongs. Each router needs a certificate for its own key, for others to use to verify the signatures it produces. Each router also needs the certificates of other routers to verify their signatures in the BGPSEC attributes it receives. Because the router certificates are part of the RPKI, they will be part of a local RPKI cache. The rpki-rtr protocol can carry the router information needed from the cache to the router. (This is presently a work in progress in the SIDR working group.)

4.2 Project Success in Standardization

The PARSONS team has produced much of the progress that has occurred in the standardization of a comprehensive BGP security solution.

At project start, there were no IETF published specifications for BGP security, although there were 9 documents that were works in progress in the SIDR working group in the IETF. The works in progress addressed origin validation only. It was not until Feb 2012 that any of the works in progress were mature enough for publication by the IETF as RFCs.

At this point, with PARSONS leadership and energetic participation, there are 24 RFCs published by the IETF from the SIDR working group. Of those 24 RFCs, 18 are co-authored by members of the PARSONS team. (See the Appendix for a list of the PARSONS team co-authored documents.) There are 17 documents that are still works in progress in the SIDR working group, of which 13 are co-authored by the PARSONS team.

Standardization of the comprehensive solution is not yet complete. Most of the progress to this point has been for origin validation. The path validation solution, which was developed by the PARSONS assembled design team, was adopted by the IETF SIDR working group as a set of related documents. Some of those documents have been published by the IETF ([23], [5]). The BGPSEC protocol specification is mature, but has not yet progressed to publication by the IETF. Furthermore, deployment experience is showing operational aspects of origin validation that may lead to further standardization activity.

4.3 Project Design Success

The project design as explained in Section 3.1 has produced strong results for the project.

4.3.1 Results: Team Multi-Stakeholder Constituency. The inclusion of participants from multiple stakeholder communities in our PARSONS team and in our assembled ad-hoc design team has proven to be an effective strategy, providing strong benefits to progress in the project.

The operation-knowledgeable members of the PARSONS team recognized that synchronizing with the global RPKI repositories and cryptographically validating the RPKI data would be a performance burden for routers in a network. The team introduced a local RPKI cache into the architecture of RPKI use in a network, in order to relieve the routers of that burden. The PARSONS team jointly designed and specified a rpki-rtr protocol for the communication between the local RPKI cache and the router. The rpki.net and the RPSTIR packages both contain implementations of the rpki-rtr protocol.

Security members of the design team identified a significant security vulnerability of a design choice early in the consideration of a path validation solution. The design choice was whether to allow or an AS to add protections to a path that was unprotected when received. The term invented for this was “partial path signing”. The security members noted that this was a privilege elevation and therefore introduced an attack possibility. If an AS invented a bogus path, then added the path validation protections to it, and claimed falsely that it had received that bogus path, the protected path might look preferable to neighbors who would be deceived.

The operators on the PARSONS team noted the very common operational practice of duplicating the local ASN multiple times in the AS path in the BGP route (“prepending”), a technique used in order to influence a neighbor’s routing decision. The protocol designers on the team were able to respond to this information and adjust the design to efficiently protect the duplicated information. The adjustment was simple - to use a counter to represent the duplication.

The path validation design team was able to identify and respond to other common practices that are deviations from the BGP standard and accommodate them in the BGPSEC design.

Operators identified a common practice at Internet exchange points, where the exchange point provides a route service that forwards BGP updates between the members. The route server does not add its AS to the path in order to avoid increasing the apparent path length. This invisible AS violates BGP path construction and would have been prevented by the path validation solution. The protocol designers on the PARSONS team were able to accommodate this behavior in the path validation protections, by employing a 0 value for the prepending counter. A potential for misuse of this feature was prevented by requiring explicit configuration to allow a neighbor’s use of a 0 counter.

Concern was raised in the working group about support in path validation for AS mergers. Organizations that acquire or merge two ASs must migrate the ASN used in their routers and their customer’s routers from one of the AS numbers to the other. The migration must take place incrementally and carefully over time. Techniques are widely implemented by router vendors that allow that migration. However, these techniques temporarily make it appear that a router is masquerading as belonging to multiple ASs, and could have been prevented by the path validation solution. Again, the protocol designers on the PARSONS team noted the problem and were able to demonstrate that the design accommodated the migration techniques.

PARSONS team members’ personal interactions with router vendor developers was an assistance to the router vendor implementation activities.

4.3.2 Results: Deployment. In Jan of 2011, individual RIRs began production service of RPKI certification of resource allocations in their regions. At this point, all RIRs are in production. This uptake by a critical member of the address allocation hierarchy indicates that the RPKI is an acceptable solution.

The take-up for the RIRs is increasing in each region. RIPE easily has the strongest membership deployment. Over 20% of their members have requested certification, and over 7000 prefixes are certified, encompassing about 4 /8 address blocks[13].

Commercial router vendors have added support for origin validation and for the rpki-rtr protocol to their router software releases. Both Cisco and Juniper have included these features in their releases, starting in 2012. Support for RPKI is available for any network operator who is using the current software releases.

Both rpki.net and RPSTIR have implemented tools to produce simple routing registry route objects from ROAs, based on a recommendation of a major RIPE ISP operator. These tools allow an easy deployment path for RPKI in a network by reusing tools for the existing best operational practice that builds filter lists from routing registries.

Deployment and use has begun, albeit barely. An Ecuador Internet Exchange Point [6] was able to work with their members and the Latin American Network Information Center, their regional RIR, to certify almost 100% of the resources in their country. They used the rpki.net software as one of two redundant relying party packages. They also announced that they would begin to employ the RPKI data to reject invalid BGP routes in their exchange. A few networks have spoken on the North American Network Operators Group (NANOG) mailing list that they have RPKI use on their networks planned for this calendar year[52, 24]. Simulation studies [51] have shown RPKI deployment at a small number of major ISPs would protect the majority of the Internet. A Lyon IXP has recently announced [1] that they have “set up an alternative filtering method of its members’ BGP route advertisements” using RPKI and ROAs. The French L’Agence Nationale de la Sécurité des Systémes d’Information (ANSSI)” has published a report ([2, 11] in which they recommend “utiliser la certification RPKI (Resource Public Key Infrastructure) et déclarer des ROA (Route Origin Authorizations),” (“use RPKI certification (Resource Public Key Infrastructure) and declare ROA (Route Origin Authorizations)”, according to an online translation service).

4.3.3 Results: Proactive Advocacy. The PARSONS team has held hands-on, real-time, workshops in every RIR region. These workshops have given the participants a full experience of using the RPKI - from requesting certification of resource, to running their own RPKI CA, to configuring Internet-connected routers to reject prefix hijacks. These workshops use the rpki.net software as both a RPKI CA and as a relying party platform.

The PARSONS team created a testbed community of early adopters who want to experiment with the RPKI but did not wish to do so with their operational networks. For this purpose, an alternative root of the address allocation hierarchy, called altCA, was created so as to issue certificates to the experimenters. Even if these certificates should be used in the global InternetSome of these early adopters had address space allocated to them from IANA before the creation of the RIRs.

PARSONS members were able to interact with communities to allay fears of the use of the RPKI. In specially noted occasions:

- Two PARSONS team members appeared in an invited panel discussing the use of the RPKI at a RIPE meeting when the membership was voting on a motion to continue or halt participation in the RPKI. The motion to continue passed.
- A PARSONS team member participated in the Federal Communications Commission (FCC) Communications Security, Reliability, and Interoperability Council III (CSRIC III) Working Group 6 - Secure BGP Deployment. The final recommendation of that working group, while not a wholesale endorsement of the use of the RPKI origin validation, includes in its recommendations [14] the desirability of resource certification and cautious, staged deployment.
- A PARSONS team member worked with the RIPE community to careful organize a series of policy proposals that would lead to the certification of legacy address space, i.e. addresses allocated before the RIRs were created [47, 40].

4.4 Results: Rpki.net

The rpki.net code is the only fully compliant implementation of both the CA and relying party parts of the comprehensive solution, including the works in progress in the SIDR working group, such as router certificates.

The rpki.net package implements features discovered to provide performance benefits. The rpki.net package implements a hierarchical file structure, which proved to have such performance benefits for rsync synchronization that other implementations have adopted that approach. It also implements parallel download of RPKI data for syncrhonization improvements, a feature suggested by the RPSTIR project.

The rpki.net implementation of the CA and RP daemons (rpkid, pubd, etc) supports multiple entities, e.g., multiple CAs supported by one publication engine. The irdbd also supports multiple entities. That means that it is possible to extend the hosted model, such as the RIRs use to provide CA services for their members, to outsourcing the publication services for any CA. The rpki.net package also implements the publication protocol [48], which provides communication between the CA issuance engines and the publication service provider.

4.4.1 RPKI Supporting Protocols. The rpki.net package has served as the platform for investigating new protocols during design and specification stages. Consequently, rpki.net includes implementations of protocols that are still works in progress.

The “out of band” setup protocol [39] was created in early testing as an improvement in boostrapping the initial relationships between parent and child engine daemons and between publication protocol clients and servers. This protocol proved so useful to early adopters and in early testing that it was brought to the SIDR working group for standardization.

The rpki.net code contains implementations of both the rpki-rtr server and the rpki-rtr client of the rpki-rtr protocol, including the enhancements to the rpki-rtr protocol in progress in the SIDR working group for communication of router public keys to the router [41].

As just mentioned, the rpki.net code contains an implementation of the publication protocol.

4.4.2 Testing Scripts. The rpki.net implementation has the ability to represent an entire test network in a single easy to understand configuration file. It has the ability to write programs which generate those test configurations, which also proved to be useful, particularly for large scale testing.

The rpki.net implementation was the test subject for modeling BitTorrent as a potential RPKI transport protocol using the Starbed large scale emulation testbed [50]. A small number of very large scale configurations used there was a valuable output of that work[36], especially in terms of improving the core code. Initial testing was done with hand-constructed test objects. The large scale testing made possible by the test generation scripts helped identify implementation issues that would otherwise have been missed.

The rpki.net implementation also includes scripts for generating large numbers of router certificates, as would be needed in any testing of BGPSEC implementations.

4.4.3 Balance Between Performance and Agile Coding. The rpki.net code uses both Python and C, with the choice made for performance reasons. Python is noted for its benefits in agile coding, and for its lower maintenance burden. This is particularly important when new protocol design choices are being explored - the faster development cycle is a help. Much of the rpki.net code originally written in C was converted to Python to adopt these advantages. However, there are places where C is needed for tasks that are performance sensitive. In rpki.net, C is used for cryptographic functions as well as the Abstract Syntax Notation One (ASN.1) and XML processing. C/Python API code was written to make the same functionalities available to the Python modules. The rcynic is the only remaining non-library C code, left in C because it needs to handle many low level details of the protocols like X.509 and Cryptographic Message Syntax (CMS) that the RPKI objects use, and the ASN.1 that underlie them.

4.4.4 Documentation and Development Environment. The database use in the rpki.net GUI uses the Django Object-Relational Mapper (ORM). This makes it simpler to change the implementation to a different implementation of SQL, e.g., SQLite instead of MySQL. This also permits use of Djang's "South" module to manage ugrades to the database schema, which minimizes impact on the users from such a change.

The rpki.net software is kept in a subversion repository, from which binary packages are built nightly for a small set of popular unix operating systems.

Documentation for the tools is written on the wiki hosted at rpki.net. The wiki pages are organized in a way that allows automatic conversion to a full document covering installation and use, for offline reading. The full manual is updated nightly and checked into the subversion repository when there are changes.

4.4.5 Rpki.net GUI. The final design of the web interface was a result of feedback received from users over the course of the project. The numerous hands on workshops allowed network operators to become familiar with the software, and give valuable feedback on how the software could be improved to support day to day workflow.

An initial example of this feedback was that the original dashboard in the web interface was primarily focused around RPKI resources. This was a natural result of the developers being focused on the RPKI itself. However, operators are more interested in a route oriented view, since this is what they are dealing with. The result of this feedback was to alter the dashboard to give primary information about currently announced routes and their route origin validity status.

After RIPE started its RPKI pilot project which allowed resources holders to host their own private cryptograhic key material, and communicate via the up/down protocol, there were a number of users starting to use the software in a semi-production manner, and this provided additional useful feedback for development. One of the first results of this was a request to have the web interface support the initial setup process between RPKI parents and children. Previously, the user needed to use the command line tool to perform this step, but many users wanted the web interface to support this feature. The process involves uploading and downloading the XML files generated during the out of band setup procedure.

One ongoing issue that is not completely solved is the lag involved in updating the web interface display of what other relying party's view of the global RPKI might be after a user makes a change to their ROAs or children's resource certificates. The creation of the ROAs and resource certificates happens nearly instantaneously, but it takes an undetermined amount of time for the objects to be published to a repository, and then subsequently fetched by a relying party and proceed into origin validation results. This is the nature of a distributed, asynchronous system, but it can be confusing to the web interface user when the origin validation on their routes does not update as soon as the user submits the changes.

Another class of user that provided useful feedback was ISPs that were interested in providing RPKI "hosting" services, where customers would not hold their own private cryptographic key materials, but could use the software to manage their own resources. This is very similar to what RIRs such as RIPE provide for their members. The result of this request was that the software added support for separating the roles of resource holder and web interface user. Instead, the model should be that a particular web interface user may want to manage resources as several different resource holders, or that multiple users could manage a single resource holding entity. This more closely models the current situation with how the RIRs manage resources, where resources for an organization may be managed by several different departments.

4.5 Results: RPSTIR

RPSTIR provides a production quality implementation of the relying party tools necessary to use the RPKI.

RPSTIR is offered under the BSD open source license model, so everyone is free to modify RPSTIR to suit individual needs or incorporate it into other products.

Features include:

- Fine-grained ASN.1-level diagnostics for debugging RPKI repositories
- Both RPSL and diagnostic output
- Top-down and bottom-up certification path discovery
- Flexible database architecture (based on MySQL)
- Efficient parallel download of RPKI objects
- Implementation of the Local Trust Anchor functionality [30] for mitigation of CA errors
- Implementation of the server for the rpki-rtr protocol
- statistics collection of the RPKI over time, including incremental updates and multiple simultaneous statistics collections.

- Support for adding files from an existing local cache. In the future, this could be used to quickly deploy additional relying party machines without requiring each to synchronize individually with the global RPKI repository system

4.6 Results: BGPSEC Implementation

The main result was the creation of functional software that implements the BGPSEC protocol specification. The software has been shown to be functional in small test environments. It can negotiate and open BGPSEC sessions between routers, send and receive BGP UPDATE messages with the BGPSEC attribute, and check that the messages are cryptographically signed correctly. While BIRD supports multiple routing protocols including BGP, the code uses a modular design for the protocols supported. This in turn allowed us to limit most of the code changes to the BGP protocol module. **Figure 3** illustrates the BIRD architecture.

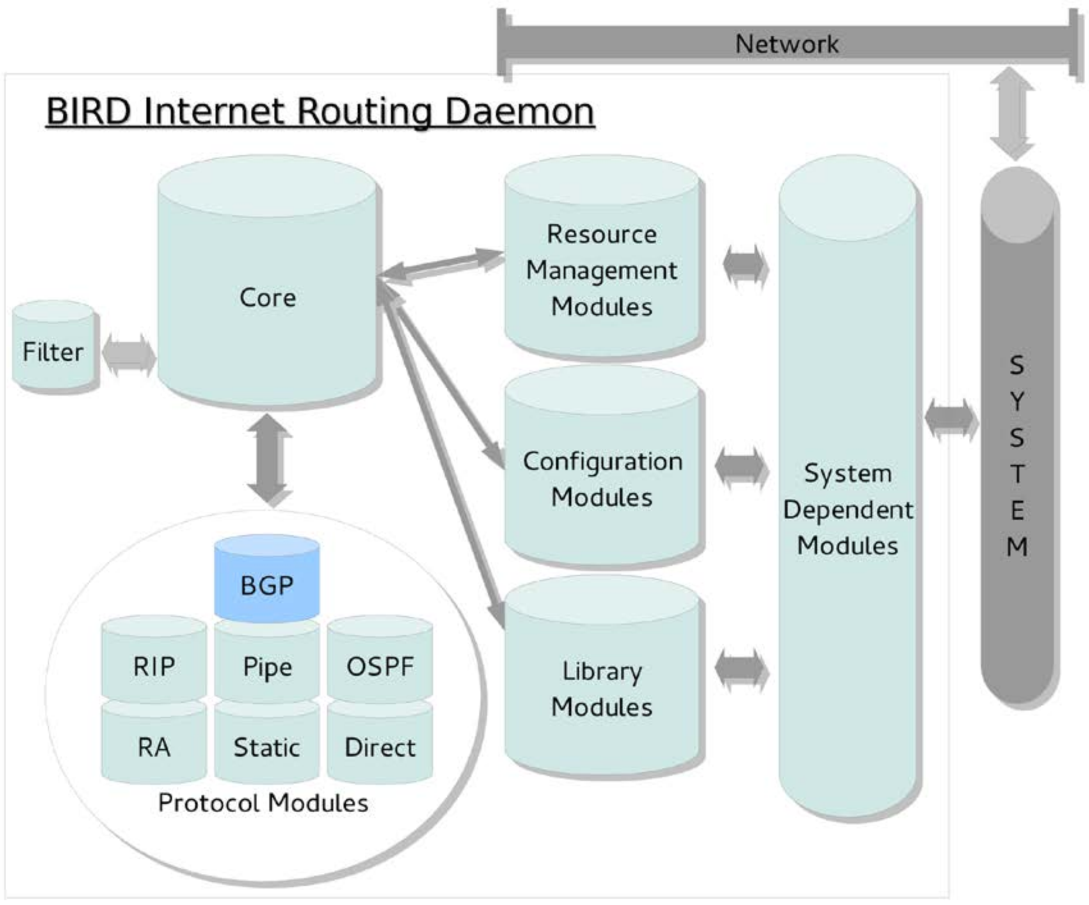

Figure 3: BIRD

BGPSEC adds security to BGP routing by providing two major protection features. One is path authentication. That is, BGPSEC cryptographically authenticates that the series of AS's that a BGP prefix announcement claims to have passed through is accurate. The other feature is network prefix origin validation. Origin validation is validating that the AS which originated a network prefix is the valid holder of that prefix.

In order to provide path authentication, an additional BGP attribute, the BGPSEC_Path attribute, is added to a BGP Update message. The BGPSEC_Path attribute contains several values related to the AS_PATH attribute of the BGP Update message, and signature values that in turn authenticate the update message and its path. In order to support BGPSEC within BIRD, we extended the BGP protocol module in BIRD's routing engine to support BGPSEC attribute handling and to support the cryptographic signing and validation of data within the BGPSEC attribute. The credentials, or keys, used to authenticate this data are garnered from the RPKI [26]. **Figure 4** shows an expansion of the BGPSEC integration into the BGP protocol module in BIRD.

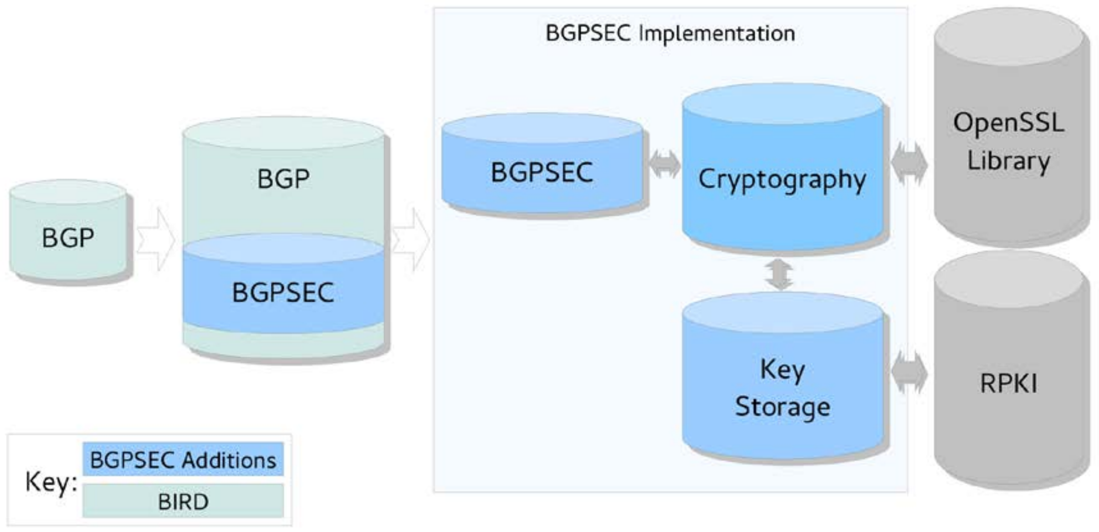

Figure 4: BGPSEC Implementation

In order to provide origin validation, the BIRD developers created an interface to create ROA tables that can be used to filter BGP prefixes based on the prefix's authorized originating AS number. The AS/Prefix data can be provided by the RPKI, but that data needs to be gathered and loaded into the ROA tables by a process outside of the BIRD daemon. RTRlib [38, 54] is an rpki-rtr client library for retrieving that data from an RPKI cache. Example client software for accessing that data was created by the RTRlib project as bird-rtrlib-cli. We in turn modified that code and provide a software client, bgpsec-bird-client, that can be used to populate the ROA tables in a running BIRD daemon.

Additionally, in order to authenticate the signatures within the BGPSEC attribute, the signing router's public keys are needed. These keys are available from the RPKI. RTRlib also provides access to these router keys. The bgpsec-bird-client software was updated to download router keys and make them available to the BGPSEC code within BIRD. An overview of the BIRD daemon with BGPSEC and bgpsec-bird-client is shown in **Figure 5** below.

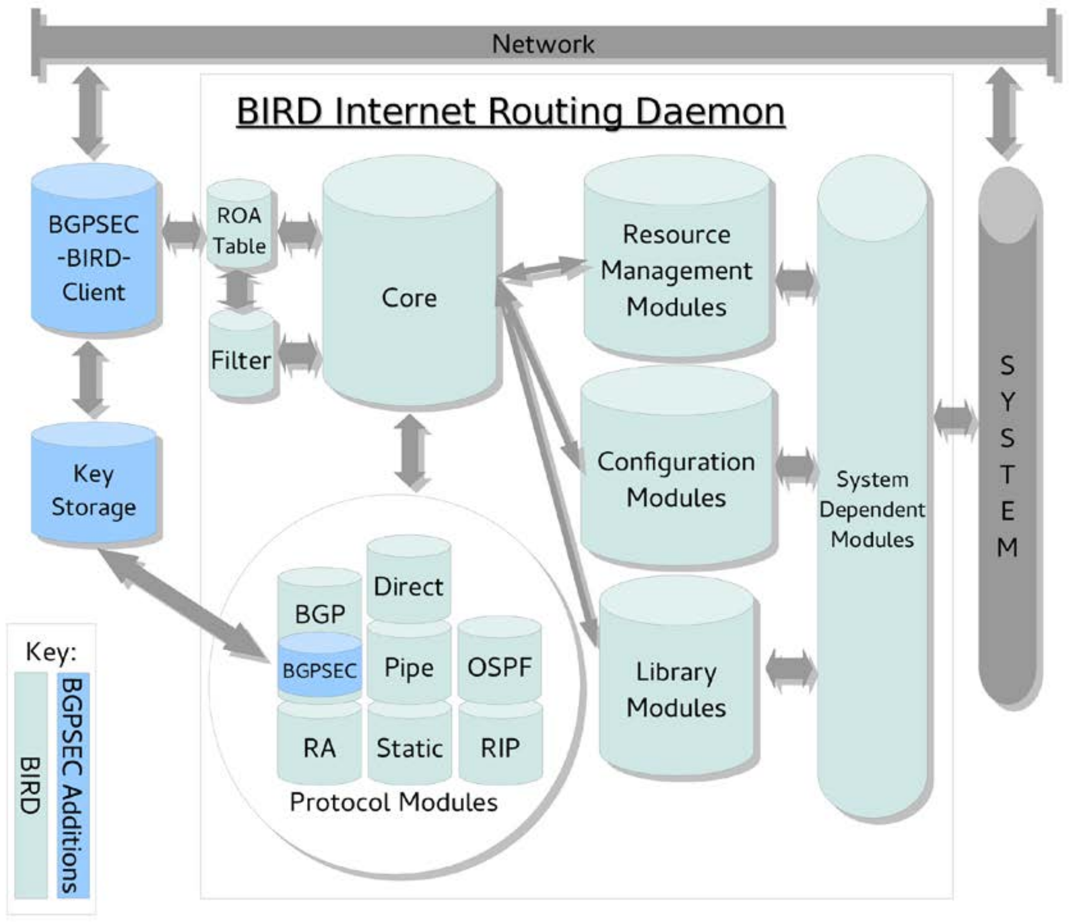

Figure 5: Overview of BGPSEC Implementation

There have been multiple releases of the software. The most current feature list follows:

- BGP capability negotiation for the use of BGPSEC in a BGP session
- Creating and parsing of the BGPSEC attribute in a BGP UPDATE message
- Generating and validating signatures within the BGPSEC attribute

- Configuration and processing of the confederation flag within the BGPSEC attribute
- Generating and processing the pcount field within the BGPSEC attribute. The pcount value is used to mimic the practice of prepending multiple copies of an AS number to an AS_PATH attribute. This is a common practice used in order to make a routing path less desirable.
- Ability to configure how Valid/Invalid BGPSEC UPDATES are treated. Local policy can choose how to make use of the Valid/Invalid states in routing decisions.
- Use of RPKI-RTR protocol [10] to get ROA data from the local RPKI cache for Origin Validation withinBIRD.
- Use of RPKI-RTR protocol [10] to get router public keys from the local RPKI cache to use for authenticating BGPSEC attribute signatures.
- ASN associated to router keys. Router keys are retrieved from the RPKI data by a combination of the router's ASN and the key identifier from the router's certificate.

4.7 Lessons Learned

4.7.1 Workshops. Workshops that provide hands-on, real-time experience with real tools are beneficial in nurturing user curiosity into experience.

The initial vision for workshops was of a training for actual deployment, with command line interface suitable for scripting, full software build and installation on a laptop, certification of all real resources managed or serviced by the participant, etc.

The experiences under this approach made it clear that operator takeup is facilitated by careful construction of the learning environment. This led to some significant changes in the workshop.

The command line interface was augmented with a GUI interface. Section 4.4.5 describes some of the lessons learned in that process.

Platform (OS, packages, versions, etc) variance is so great that building and installing software takes a large amount valuable time. Workshop environments must provide a simplified environment in which unessential details are invisible and participants focus only on the essential topics. More effort was put into producing binary packages for popular platforms to reduce the effort needed just to participate.

The state of the prefix allocation system is such that it is difficult for some major long-time operators to know what prefixes they have been allocated and what prefixes have been suballocated to others and to whom. Rather than have the participants attempt to use resources for which they are responsible, An experimental allocation was request from ARIN for use as a workshop subject.

The intent of the workshops changed from an expectation of full certification of all real resources as if for immediate operational deployment to an opportunity to educate and entice the operators, improve the rpki.net's parallel with typical operator practice, and improve the robustness of the software to odd network environments.

4.7.2 Testing and Deployment Tools. Quick prototype deployment tools can themselves become targets for standardization. We experienced this both in the design of the rpki-rtr protocol and in the out-of-band setup protocol. We also experienced this in tools that proved to be so useful that they were incorporated into the web GUI for rpki.net.

4.7.3 Revisiting Design Decisions as Capabilities are Added. At multiple points in our implementation history, we discovered that when adding a new capability to the system, it is wise to revisit past design decisions to determine if those design decisions are still valid in the new implementation makeup.

Our introduction of a web server was such a case. Initially our code was designed not to require an external web server. This was no longer the case once we added the Django-based user interface to the design. In retrospect, we could have reconsidered a few aspects of the overall implementation design. In particular, there are some tasks that could be simplified using the off-the-shelf web proxy module that comes with the web server we're using. It would have let us prune some unnecessary features out of the code we've been maintaining, simplifying future development and maintenance, and might have led us earlier to a more robust design for users who want redundant server configurations.

The decision to add the Django user interface also brought the Django ORM into use in our code. In retrospect, we could have converted all of our existing home-grown SQL interface code to use Django's ORM. Using the ORM has two advantages. First it allows easier migration to different SQL implementations. Second, the Django "South" add-on module, which is a tool for managing SQL schema upgrades, is intended to work with the Django ORM.

At the time, an interest in maintaining the stability of the code and avoid a difficult upgrade for the users convinced us to retain the existing interface, despite the potential easing of future maintenance and development. However, it has been necessary to modify the SQL schema since that decision was made, and the South add-on module would have eased that task.

Decisions made for keeping the code base stable are sometimes good, sometimes regretted later.

Reusing standard libraries is good. Revisiting old decisions not to reuse standard libraries is also good: in some cases, one can only really understand why some standard library behaves in the odd way it does by following at least part way along the implementation path that led to the strange library behavior, at which point it becomes obvious.

We use lots of standard library code. We also re-implement some things when the available libraries are clearly insufficient for our needs. The lesson learned here is just that it's good to go back and reexamine these decisions every few years, as the decision can change with time.

4.7.4 Testing. Interoperability testing, even if informal, is beneficial. There was cross-fertilization between the rpki.net and RPSTIR implementations. The rpki.net choice to store the RPKI data in a hierarchical structure proved to provide better performance, and other implementations eventually followed suit. The RPSTIR choice to use parallel retrievals of RPKI data also proved to have performance benefits. Operational experience with rsync led to the conclusion that parallel fetches would greatly improve the time required for a RP to mirror repositories. This was adopted by other implementations, including rpki.net.

Interoperability testing between the known relying party implementations using the RPSTIR compliance tests proved beneficial to all parties. Some tests exposed ambiguities in the specification, resulting in improvements in the specification. Other tests led to improvements in the implementations.

Large scale testing (at the Starbed facility) is very useful for software that will be deployed at Internet scale. The rpki.net implementation served as a test subject for an emulation experiment testing BitTorrent's suitability as a RPKI transport protocol. Since the need to replace rysnc as a transport protocol was in large part due to its inability to scale, this experiment tested BitTorrent's behavior at scale. The large scale test configurations developed for that experiment exposed implementation issues related to the sheer scale of the test. The initial tests had been hand-configured scenarios which were useful at the time, but which did not stress the implementation to expose those sorts of issues.

The ability to represent an entire test network in a single easy to understand configuration file proved useful from the beginning of the rpki.net development. The ability to write programs which generate those test configurations also turned out to be useful. The large scale test configurations that were a by-product of the emulation experiment were the experiments chief benefit, from the viewpoint of improving the code.

4.7.5 Tradeoffs between Performance and Agile Coding. Python has a lot of advantages for quick development: it's much less fragile than C, and much more concise, so one screen of Python code can take the place of ten or twenty times as much C code. It is also easier to debug and maintain. However, while Python is reasonably fast for a higher level language, it is not as fast as C at certain tasks. Using C for the actual cryptography was an obvious choice: what was less obvious was that relatively straightforward data encoding in both ASN.1 and XML was highly performance sensitive and really needed to be done in C. We wrote ASN.1 code in C and then wrote C/Python API code to make this functionality available to Python.

We learned that the right approach was to prototype in Python, profile to find the hot spots, and convert to C as necessary for performance.

Maintenance effort can make conversion in the other direction the right approach for some tasks. Some code written in C in the early days of the project turned out to be more trouble than it was worth to maintain. Using the same library code developed for the core protocol daemons, we were able to rewrite all of these legacy C programs as relatively trivial Python scripts.

At times the balance between performance and agility is not clear. For example, the RPKI validation engine (rcynic) is still written in C. Because it must deal with the many small details of the X.509, CMS and ASN.1 technologies that underlie the RPKI, an attempt to rewrite this in Python would be unwieldy. However, a Python implementation would provide a faster development cycle for experimentation during protocol design.

4.8 Challenges and Residual Issues

4.8.1 Transport Protocol. The choice to concentrate on security architecture and to choose a well-known existing transport protocol allowed concentration on the crucial security requirements during the solution design and implementation. The transport protocol chosen, rsync, has served well for initial deployment but proved to have some drawbacks.

The design of the RPKI treated the publication transport problem as a black box, and chose rsync as a widely used, well understood protocol. rsync provided a pull-based transport, requiring only deltas to be retrieved. Because the RPKI is object security, i.e., the security does not depend on the channel through which you receive the data, there was an opportunity for architectures that would distribute data through neighbors, rather than mandate retrieval directly from an central server. This significantly reduced the scope of the initial implementation problem, which allowed us to focus on core security architecture issues.

However, there were some attributes of rsync that presented unexpected problems

- Transactional behavior in publication: Changes to a repository's publication point should be made as one atomic action, so the manifest is always current with the contents. But rsync has no transactional features.
- Server side scaling issues: rsync recalculates on the fly. This is a benefit to the server side implementer who doesn't have to worry about those issues, but it scales very badly for the operator of the server. There were concerns that eventually deployment would reach a point that servers, particularly those providing a hosted CA service, would experience capacity problems. That point has not been reached yet and is not close, so now is the time to consider an alternative transport protocol, before the need is urgent.

A change in the transport protocol will not change the security solution. The security architecture is not reliant upon features of the transport protocol, so its design will not be perturbed by a change in the transport protocol. Other implementation decisions made on the basis of the transport protocol may themselves change. For example, because rsync is file based, the retrieved RPKI data in rpki.net is kept in a file system structure. This will likely change to a true database should the transport protocol change to one that is not file based.

4.8.2 Legacy Space. The address allocation system as described before is a hierarchy. The hierarchy originally was quite broad - the IANA allocated addresses directly to recipients. When the RIR system was established, addresses that were allocated directly by IANA were termed “legacy address space”. The legacy address space holder may or may not be members of their region’s RIR. This means that the RIR may not be willing to certify their address space. As reported, one policy proposal adopted in RIPE [47] allows certification of the legacy address space in the RIPE region. However, the other regions have not followed suit. The RIPE implementation of the policy created a fee for the service that might hamper even that effort.

Legacy space represents a large portion of some region’s addresses in use. In particular, of the organizations served by ARIN, more than 40% hold legacy address space [28]

If that address space can not be certified by RIR policy, then there is still considerable risk to the Internet users of attacks through the routing infrastructure for that space.

This problem may lessen in this period of IPv4 run-out. Legacy addresses are IPv4 addresses. As the market for IPv4 addresses becomes stronger, it is anticipated that those who hold legacy space will transfer their address space to others in the transfer market. The result of a formal transfer is that the new holder will enter into a regular member agreement with ARIN and will not longer be considered a legacy space holder.

4.8.3 Non-Technical Concerns and External Influences. There are several external, non-technical forces and concerns that recur in discussions frequently.

- Fears of a hierarchy continue. Although the current address allocation system is hierarchical, swift enforcement of that hierarchy is difficult. The provision of a secure hierarchy represents the potential that the hierarchy could now be enforceable. This raises fears that the hierarchy could be misused to impact routing operations - through error, through business issues, through legal and outside forces, etc.
- Introduction of any new technology represents risk and effort. As with most security systems, the return on investment for this new technology is not immediate, or clear, or even visible.
- Errors a CA makes could lead to harm to someone’s routing, causing them damage. This has led to a concern about legal liability. This concern is behind a movement by many RIRs in the SIDR working group to soften the enforcement of certification [17], to lessen the chance that they would be responsible for damage. The concern over legal liability also has led ARIN to insist that a relying party seeking access to the ARIN RPKI data must agree first to a Relying Party Agreement (RPA). This RPA forbids sharing of the data, which limits some of the architectural distribution methods that are technically possible and forces everyone to retrieve the ARIN RPKI data directly from ARIN.
- Some operators are concerned that the top of this hierarchy at the RIRs are not accustomed to an operational role in routing and may not execute that role with stability and robustness.

That could hamper the operator's ability to meet its own SLA agreements with its customers.

4.8.4 RPKI Latency. The present best practice for origin validation relies on Internet Routing Registries (IRRs) who provide a routing registry service. Registered route objects are retrieved from these known sites, typically daily, and used to form filters on the BGP updates. There is a lag in this system between registration of new information and the time it is retrieved and put into a filter. The RPKI would behave in a similar pull fashion, and would experience a similar latency between the time a ROA is published and the time it is retrieved and used in a router.

There are some business models that would eventually require the ability to send authorized BGP announcements on a moment's notice. The RPKI latency as relying parties retrieve the information on their own schedule would interfere with that business model. The current best practice would also interfere in the same way, but the desire is that the RPKI be capable of improving that situation.

The latency should be a requirement for consideration in any choice of a new transport protocol.

4.8.5 BGPSEC Islands of Trust. Avoiding a privilege escalation by disallowing partial path signing means that a BGP router can use the BGPSEC path validations only with BGPSEC capable neighbors, and only for routes it received from a BGPSEC capable neighbor or originated itself. Adoption will grow outward organically from individual first adopters, as islands of trust.

4.8.6 Routing Problems outside the BGP Protocol. There are some problems in the operation of the routing infrastructure that do not involve BGP mis-behavior. Some operators are not interested in a security solution to BGP if it does not solve the non-BGP mis-behavior.

Principal among these issues is route leaks.

A route leak is a neighbor's violation of an understood constraint on the propagation of an update that you have sent to the neighbor. This is frequently a matter of the business relationships. A transit provider sends updates to a customer, but expects that customer not to propagate to any other networks. Or a peer may send routes to its customers to a peer, but he expects the peer to propagate the update only within its customer cone. Sometimes the constraint is not business relationship based, for example, if a set of ASs wish to form a virtual community and agree not to send updates to ASs that are outside the community for certain updates.

This constraint is implied - it is not expressed in the open, it does not appear in BGP, it is not enforced by BGP, it is not part of defined BGP behavior.

In order that a BGP security solution should protect against this mis-behavior, the constraint has to be expressed, available in the open, perhaps in BGP, and BGP defined behaviour would have to be modified to enforce the constraint.

In recognition of the interest in this problem, the SIDR, GROW, and IDR working groups in the IETF devised a plan of action. GROW would address a definition of the problem, IDR would make any necessary changes to the BGP protocol, and SIDR would consider protections for the new features of BGP.

That plan is underway. The GROW working group has adopted route leaks as a work item [12] and two proposals have been made [4, 25] for extensions to bgpsec that would securely express and transmit the information needed to stop route leaks.

4.8.7 Uncertain Address Records. Larger ISPs, especially those who have a long history, perhaps back to legacy address days, find that their records of what addresses they hold and what addresses they have allocated to customers have not been well maintained. This makes it necessary for them to preform an audit of their address allocation records before they can know what certificates and ROAs they should issue or request. An incorrect or unwise certificate or ROA could cause a customer's BGP route to be judged invalid, so care is needed.

4.8.8 Further Specification Development. As operational experience progresses, further specification of tools or protocols may be needed, e.g., specification of a replacement transport protocol. Work has already begun on specifications not yet adopted by a working group to specify the process for transfer of an address prefix to a new holder [15] and to provide a fail safe mechanism to help identify inappropriate behavior by a superior in the RPKI hierarchy [46].

5 CONCLUSIONS

The result of this effort has had a visibly positive impact on the standardization, implementation and initial deployment of the RPKI solution for origin validation, and the design, standardization, and implementation of a path validation solution. The majority of the specifications published or underway within the IETF were co-authored by PARSONS and their team members. Word about the problem space and coming solution space has been spread within multiple operational communities. The first implementations of solution components were produced by PARSONS and their team members. Clearly, without this effort the state of deployment would be hugely reduced and likely entirely non-existent.

The road to the current state of deployment has had its fair share of rough spots and has seen the occasional unexpected pothole. The summary of the results of the project along with unexpected major stumbling blocks, along with the corresponding lessons learned, are listed in the sections below.

5.1 Summary of Results

The following describes the high-level aggregated results of this effort, both positive and negative.

- **Positive:** *Standardization efforts significantly advanced*

  The standardization efforts of documenting both the RPKI and BGPSEC within the IETF standardization body has progressed significantly well. The initial documents describing the RPKI framework and origin validation have been published as RFCs and the BGPSEC path validation specifications are nearing completion of work and the start of the publication process.

- **Positive:** *Operational communities aware of the routing security efforts through outreach efforts*

  Through presentations made by PARSONS team members, operational communities worldwide have been made aware of the routing security efforts and the related RPKI and BGPSEC solution components. Workshops and demonstrations have been given to help train operators about what they can expect when full deployment begins.

- **Positive:** *Implementations of RPKI and BGPSEC are available*

  Multiple implementations of RPKI certification and operational use are freely available; sophisticated compliance testing suites for certification authorities are available

- **Positive:** *Router vendors have implemented RPKI use*

  Major router vendors have implemented RPKI use and configuration in their major software releases

- **Negative:** *Operator reluctance still impacting deployment*

  Unfortunately, the operational staffing is still heavily impacted with day-to-day operations and extra workload placed on them in order to secure the routing infrastructure still appears daunting to them. Because the threats are not perceived as immediate, the timeline for rolling out deployment is not deemed to be immediate.

- **Positive:** *Policy to certify legacy space adopted in RIPE*

With the active involvement of the PARSONS team, the RIPE community has adopted a policy that would allow legacy address holders to receive RPKI certification of their resources

- **Negative:** *Concern over misuse of hierarchical authority in the RPKI continues*

  Operators continue to be concerned about damage on routing operations resulting from deliberate, accidental, or forced actions on the part of the RPKI hierarchy.

- **Negative:** *Concern over RIR ability to operate a robust, stable, globally and continually available service*

  The RIRs have a critical position in the RPKI hierarchy that requires a level of service beyond what they are used to providing. Failure to provide that level of service could impact the ability of their members to meet their own service level agreements.

5.2 Final Status

The current status of the project is:

standards produced

18 of the 24 IETF publications defining the RPKI, origin validation and path validation are co-authored by the PARSONS team.

implementations produced

Three different implementation packages have been produced:

1. The only open source RPKI implementation of both CA certification and relying party operational use of the RPKI that is fully compliant with IETF publications and with work in progress,
2. a production level implementation of the relying party operational use of the RPKI, a suite of compliance cases to test CA output, and an implementation of the local trust anchor management specification [30],
3. and a functional implementation of the BPGSEC protocol.

deployment announcements

First announcements have been made that RPKI operational use in routing has begun or is planned for this year. Some governments have recommended certification of resources in general or in the RPKI in particular.

continuing work

As deployment experience continues, enhancements and corrections to existing specifications and implementations will be needed.

tests and experiments Major ISPs are known to be testing in their own internal labs, and others are making use of the testing environment and alternate trust anchor, altCA, that is made available by this project team.

6 REFERENCES

[1] Nouvelle methode de filtrage sur les route serveurs : Rpki + roa, October 2014. http://www.lyonix.net/en/media-adnix/archives/item/nouvelle-methode-de-filtrage-sur-les-route-serveurs-rpki-roa-9.

[2] ANSSI. L'observatoire de la resilience de l'internet francais publie son rapport 2013. Communique de presse, Sep 2014. http://www.ssi.gouv.fr/IMG/pdf/CP-OBSERVATOIRE-RAPPORT-2013.pdf.

[3] R. Austein, G. Huston, S. Kent, and M. Lepinski. Manifests for the Resource Public Key Infrastructure (RPKI). RFC 6486, February 2012.

[4] B. Dickson (editor). Route Leaks – Proposed Solutions. IETF Internet-Draft, March 2012. https://tools.ietf.org/html/draft-dickson-sidr-route-leak-solns-01.

[5] S. Bellovin, R. Bush, and D. Ward. Security Requirements for BGP Path Validation. RFC 7353, August 2014.

[6] Sofia Silva Berenguer. Rpki and origin validation deployment in ecuador, November 2013. http://iepg.org/2013-11-ietf88/RPKI-Ecuador-Experience-v2b-1.pdf.

[7] BIRD Internet Routing Daemon. http://bird.network.cz.

[8] R. Bush. The Resource Public Key Infrastructure (RPKI) Ghostbusters Record. RFC 6493, February 2012.

[9] R. Bush. Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI). RFC 7115, January 2014.

[10] R. Bush and R. Austein. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810, January 2013.

[11] Francois Contat, Mathieu Feuillet, Peter Lorinquer, Samia M'timet, Guillaume Valadon, Remi Varloot, and Nicolas Vivet. Resilience de l'internet francais. Technical report, L'Agence nationale de la securite des systemes d'information (ANSSI), 2013. http://www.ssi.gouv.fr/IMG/pdf/Rapport_Observatoire_2013.pdf.

[12] Danny McPherson and Shane Amante and Eric Osterweil and Dave Mitchell. Route-Leaks & MITM Attacks Against BGPSEC. IETF GROW Working Group Internet Draft, April 2014. https://tools.ietf.org/html/draft-ietf-grow-simple-leak-attack-bgpsec-no-help-04.

[13] Andrew de la Haye. Ripe ncc update, April 2014. https://www.arin.net/participate/meetings/reports/ARIN_33/mem_transcript.html_#anchor_3.

[14] CSRIC III Working Group 6 Secure BGP Deployment. Working group 6 secure bgp deployment final report, September 2012. http://transition.fcc.gov/bureaus/pshs/advisory/csric3/CSRICIII_9-12-12_WG6-Final-Report.pdf.

[15] Edric Barnes. Resource Public Key Infrastructure (RPKI) Resource Transfer Protocol and Transfer Authorization Object (TAO). IETF SIDR Working Group Internet Draft, February 2014. http://tools.ietf.org/html/draft-barnes-sidr-tao.

[16] R. Gagliano, S. Kent, and S. Turner. Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI). RFC 6916, April 2013.
[17] Geoff Huston and George Michaelson and Carlos Martinez and Tim Bruijnzeels and Andrew Newton and Alain Aina,. RPKI Validation Reconsidered. IETF SIDR Working Group Internet Draft, July 2014. `https://draft-ietf-sidr-rpki-validation-reconsidered`.
[18] G. Huston, R. Loomans, B. Ellacott, and R. Austein. A Protocol for Provisioning Resource Certificates. RFC 6492, February 2012.
[19] G. Huston, R. Loomans, and G. Michaelson. A Profile for Resource Certificate Repository Structure. RFC 6481, February 2012.
[20] G. Huston, G. Michaelson, and S. Kent. Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI). RFC 6489, February 2012.
[21] G. Huston, G. Michaelson, and R. Loomans. A Profile for X.509 PKIX Resource Certificates. RFC 6487, February 2012.
[22] IETF. The Internet Engineering Task Force. `http://www.ietf.org`.
[23] S. Kent and A. Chi. Threat Model for BGP Path Security. RFC 7132, February 2014.
[24] Jac Kloots. Bgpmon alert questions, April 2014. `http://mailman.nanog.org/pipermail/nanog/2014-April/066071.html`.
[25] Kotikalapudi Sriram and Doug Montgomery, editors. Enhancement to BGPSEC for Protection against Route Leaks. IETF Individual Submission Internet Draft, July 2014. `https://tools.ietf.org/doc/<draft-sriram-route-leak-protection`.
[26] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480, February 2012.
[27] M. Lepinski, S. Kent, and D. Kong. A Profile for Route Origin Authorizations (ROAs). RFC 6482, February 2012.
[28] Leslie Nobile. Registration Services Update. ARIN 31 Meeting (Presentation), April 2013. `https://www.arin.net/participate/meetings/reports/ARIN_31/PDF/wednesday/nobile_rsd.pdf`.
[29] C. Lynn, S. Kent, and K. Seo. X.509 Extensions for IP Addresses and AS Identifiers. RFC 3779, June 2004.
[30] M. Reynolds and S. Kent (editors). Local Trust Anchor Management for the Resource Public Key Infrastructure. IETF SIDR Working Group Internet Draft, April 2014. `http://tools.ietf.org/html/draft-ietf-sidr-ltamgmt`.
[31] M. Reynolds, S. Turner and S. Kent (editors). A Profile for BGPSEC Router Certificates, Certificate Revocation Lists, and Certification Requests. IETF SIDR Working Group Internet Draft, August 2014. `https://tools.ietf.org/html/draft-ietf-sidr-bgpsec-pki-profiles-08`.
[32] Matthew Lepinski (editor). BGPSEC Protocol Specification. IETF SIDR Working Group Internet Draft, September 2014. `https://tools.ietf.org/doc/draft-ietf-sidr-bgpsec-protocol/`.
[33] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. BGP Prefix Origin Validation. RFC 6811, January 2013.
[34] David Clark, Thomas Berson and Herbert S. Lin, Editors; Committee on Developing a Cybersecurity Primer: Leveraging Two Decades of National Academies Work; Computer Science and Telecommunications Board; Division on Engineering and Physical Sciences;

National Research Council. *At the Nexus of Cybersecurity and Public Policy: Some Basic Concepts and Issues*. The National Academies Press, 2014.
[35] Office of the President George Bush. The national strategy to secure cyberspace, February 2003. `https://www.us-cert.gov/sites/default/files/publications/cyberspace_strategy.pdf`.
[36] Debbie Perouli, Olaf Maennel, Iain Phillips, Sonia Fahmy, Randy Bush, and Rob Austein. An Experimental Framework for BGP Security Evaluation. *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 55(4):147–154, 2013.
[37] The Django Project. The django web framework. `www.djangoproject.com`.
[38] Joint project of the INET research group at the Hamburg University of Applied Sciences and the CST research group at Freie UniversitÃ¤t Berlin. Rtrlib - the rpki rtr client c library, June 2014. `rpki.realmv6.org`.
[39] R. Austein (editor). An Out-Of-Band Setup Protocol For RPKI Production Services. IETF SIDR Working Group Internet Draft, July 2014. `https://tools.ietf.org/html/draft-ietf-sidr-rpki-oob-setup`.
[40] R. Bush. Introduction to Legacy Services, PI Services, & Certifying Them. RIPE 66 Meeting (Presentation), May 2013. `https://ripe66.ripe.net/presentations/208-130515.pi-legacy-intro.pdf`.
[41] R. Bush and R. Austein (editors). The Resource Public Key Infrastructure (RPKI) to Router Protocol. IETF SIDR Working Group Internet Draft, August 2014. `https://tools.ietf.org/html/draft-ietf-sidr-rpki-rtr-rfc6810-bis`.
[42] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918, February 1996.
[43] RIPE. Youtube hijacking: A ripe ncc ris case study. `http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study`.
[44] University of oregon route views archive project. `http://routeviews.org`.
[45] rpki.net. Rpki tools manual, October 2014. `http://subvert-rpki.hactrn.net/trunk/doc/manual.pdf`.
[46] S. Kent and D. Mandelberg, editors. Suspenders: A Fail-safe Mechanism for the RPKI. IETF Individual Submission Internet Draft, July 2014. `https://tools.ietf.org/doc/draft-kent-sidr-suspenders`.
[47] S. O'Reilly, B. Tuy, D. Wilson, S. Steffan, H. Eidnes, H. Nussbacher, C. Friacas and R. Bush. RIPE NCC Services to Legacy Internet Resource Holders. RIPE Policy Proposal 2012-07 v4.0, October 2013. `https://www.ripe.net/ripe/policies/proposals/2012-07`.
[48] S. Weiler, A. Sonalker, and R. Austein (editors). A Publication Protocol for the Resource Public Key Infrastructure (RPKI). IETF SIDR Working Group Internet Draft, February 2014. `https://tools.ietf.org/html/draft-ietf-sidr-publication`.
[49] SIDR. Secure Inter-Domain Routing, IETF Working Group. `https://datatracker.ietf.org/wg/sidr/`.
[50] Starbed large scale emulation testbed. `http://starbed.nict.go.jp/en/`.

[51] Alexandru Stefanescu. Effects of rpki deployment on bgp security, August 2011. `http://www.caida.org/workshops/bgp-traceroute/slides/bgp-traceroute1108_rpki_deployment_study.pdf`.
[52] Mark Tinka. Bgpmon alert questions, April 2014. `http://mailman.nanog.org/pipermail/nanog/2014-April/065989.html`.
[53] W3C. World Wide Web Consortium (W3C). `http://www.w3.org/`.
[54] Matthias Wählisch, Fabian Holler, Thomas C. Schmidt, and Jochen H. Schiller. Rtrlib: An open-source library in c for rpki-based prefix origin validation. In *Presented as part of the 6th Workshop on Cyber Security Experimentation and Test*, Washington, D.C., 2013. USENIX.

APPENDIX: PARSONS CO-AUTHORED IETF DOCUMENTS

The following is a list of the published IETF documents that are co-authored by the PARSONS team.

A.1 IETF Published RFCs

- Lepinski, M. and S. Kent, “An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012, <http://www.rfc-editor.org/info/rfc6480>.
- Lepinski, M., Kent, S., and D. Kong, “A Profile for Route Origin Authorizations (ROAs)", RFC 6482, February 2012, <http://www.rfc-editor.org/info/rfc6482>.
- Kent, S., Kong, D., Seo, K., and R. Watro, “Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)", BCP 173, RFC 6484, February 2012, <http://www.rfc-editor.org/info/rfc6484>.
- Austein, R., Huston, G., Kent, S., and M. Lepinski, “Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, February 2012, <http://www.rfc-editor.org/info/rfc6486>.
- Lepinski, M., Chi, A., and S. Kent, “Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, February 2012, <http://www.rfc-editor.org/info/rfc6488>.
- Huston, G., Michaelson, G., and S. Kent, “Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, February 2012, <http://www.rfc-editor.org/info/rfc6489>.
- Huston, G., Weiler, S., Michaelson, G., and S. Kent, “Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", RFC 6490, February 2012, <http://www.rfc-editor.org/info/rfc6490>.
- Manderson, T., Vegoda, L., and S. Kent, “Resource Public Key Infrastructure (RPKI) Objects Issued by IANA", RFC 6491, February 2012, <http://www.rfc-editor.org/info/rfc6491>.
- Huston, G., Loomans, R., Ellacott, B., and R. Austein, “A Protocol for Provisioning Resource Certificates", RFC 6492, February 2012, <http://www.rfc-editor.org/info/rfc6492>.
- Bush, R., “The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, February 2012, <http://www.rfc-editor.org/info/rfc6493>.
- Bush, R. and R. Austein, “The Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6810, January 2013, <http://www.rfc-editor.org/info/rfc6810>.
- Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, “BGP Prefix Origin Validation", RFC 6811, January 2013, <http://www.rfc-editor.org/info/rfc6811>.
- Gagliano, R., Kent, S., and S. Turner, “Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI)", BCP 182, RFC 6916, April 2013, <http://www.rfc-editor.org/info/rfc6916>.

- Bush, R., Wijnen, B., Patel, K., and M. Baer, “Definitions of Managed Objects for the Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6945, May 2013, <http://www.rfc-editor.org/info/rfc6945>.
- Bush, R., “Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)", BCP 185, RFC 7115, January 2014, <http://www.rfc-editor.org/info/rfc7115>.
- Bush, R., Austein, R., Patel, K., Gredler, H., and M. Waehlisch, “Resource Public Key Infrastructure (RPKI) Router Implementation Report", RFC 7128, February 2014, <http://www.rfc-editor.org/info/rfc7128>.
- Kent, S. and A. Chi, “Threat Model for BGP Path Security", RFC 7132, February 2014, <http://www.rfc-editor.org/info/rfc7132>.
- Bellovin, S., Bush, R., and D. Ward, “Security Requirements for BGP Path Validation", RFC 7353, August 2014, <http://www.rfc-editor.org/info/rfc7353>.

A.2 IETF Internet-Drafts, Works in Progress

The following IETF internet-drafts, co-authored by members of the PARSONS team, have been adopted by the IETF SIDR working group as work items.

- Matt Lepinski, “BGPSEC Protocol Specification", draft-ietf-sidr-bgpsec-protocol-09.txt, 2014-07-04, <http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-protocol-09.txt>

  This document describes BGPSEC, an extension to the Border Gateway Protocol (BGP) that provides security for the path of autonomous systems through which a BGP update message passes. BGPSEC is implemented via a new optional non-transitive BGP path attribute that carries a digital signature produced by each autonomous system that propagates the update message.
- Matt Lepinski, Sean Turner, “An Overview of BGPSEC", draft-ietf-sidr-bgpsec-overview-05.txt, 2014-07-04, <http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-overview-05.txt>

  This document provides an overview of a security extension to the Border Gateway Protocol (BGP) referred to as BGPSEC. BGPSEC improves security for BGP routing.
- Mark Reynolds, Sean Turner, Stephen Kent, “A Profile for BGPSEC Router Certificates, Certificate Revocation Lists, and Certification Requests", draft-ietf-sidr-bgpsec-pki-profiles-08.txt, 2014-08-12, <http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-pki-profiles-08.txt>

  This document defines a standard profile for X.509 certificates for the purposes of supporting validation of Autonomous System (AS) paths in the Border Gateway Protocol (BGP), as part of an extension to that protocol known as BGPSEC. BGP is a critical component for the proper operation of the Internet as a whole. The BGPSEC protocol is under development as a component to address the requirement to provide security for the BGP protocol. The goal of BGPSEC is to design a protocol for full AS path validation based on the use of strong cryptographic primitives. The End-Entity (EE) certificates specified by this profile are issued under Resource Public Key Infrastructure (RPKI) Certification Authority (CA) certificates, containing the AS Identifier Delegation extension, to routers within the Autonomous System (AS). The certificate asserts that the router(s) holding the private key are authorized to send out secure route advertisements on behalf of the specified AS. This document also profiles the Certificate Revocation List (CRL), profiles the format of certification requests, and specifies Relying Party certificate path validation procedures. The document extends the RPKI; therefore, this documents updates the RPKI Resource Certificates Profile (RFC 6487).

- Sean Turner, Keyur Patel, Randy Bush, "Router Keying for BGPsec", draft-ietf-sidr-rtr-keying-07.txt, 2014-05-23, <http://tools.ietf.org/html/draft-ietf-sidr-rtr-keying-07.txt>

  BGPsec-speaking routers are provisioned with private keys to sign BGP messages; the corresponding public keys are published in the global RPKI (Resource Public Key Infrastructure) thereby enabling verification of BGPsec messages. This document describes two ways of provisioning the public-private key-pairs: router-driven and operator-driven.

- Stephen Kent, Derrick Kong, Karen Seo, "Template for a Certification Practice Statement (CPS) for the Resource PKI (RPKI)", draft-ietf-sidr-cps-04.txt, 2014-04-28, <http://tools.ietf.org/html/draft-ietf-sidr-cps-04.txt>

  This document contains a template to be used for creating a Certification Practice Statement (CPS) for an Organization that is part of the Resource Public Key Infrastructure (RPKI), e.g., a resource allocation registry or an ISP.

- Wesley George, Sandra Murphy, "BGPSec Considerations for AS Migration", draft-ietf-sidr-as-migration-02.txt, 2014-07-29, <http://tools.ietf.org/html/draft-ietf-sidr-as-migration-02.txt>

  This draft discusses considerations and methods for supporting and securing a common method for AS-Migration within the BGPSec protocol.

- Rob Austein, "An Out-Of-Band Setup Protocol For RPKI Production Services", draft-ietf-sidr-rpki-oob-setup-01.txt, 2014-07-02, <http://tools.ietf.org/html/draft-ietf-sidr-rpki-oob-setup-01.txt>

  This note describes a simple out-of-band protocol to ease setup of the RPKI provisioning and publication protocols between two parties. The protocol is encoded in a small number of XML messages, which can be passed back and forth by any mutually agreeable secure means.

This setup protocol is not part of the provisioning or publication protocol, rather, it is intended to simplify configuration of these protocols by setting up relationships and exchanging BPKI keying material.

- Randy Bush, “RPKI Local Trust Anchor Use Cases", draft-ietf-sidr-lta-use-cases-01.txt, 2014-06-28, <http://tools.ietf.org/html/draft-ietf-sidr-lta-use-cases-01.txt>

  There are a number of critical circumstances where a localized routing domain needs to augment or modify its view of the Global RPKI. This document attempts to outline a few of them.

- Randy Bush, Rob Austein, “The Resource Public Key Infrastructure (RPKI) to Router Protocol", draft-ietf-sidr-rpki-rtr-rfc6810-bis-02.txt, 2014-08-29, <http://tools.ietf.org/html/draft-ietf-sidr-rpki-rtr-rfc6810-bis-02.txt>

  In order to verifiably validate the origin Autonomous Systems and Autonomous System Paths of BGP announcements, routers need a simple but reliable mechanism to receive Resource Public Key Infrastructure (RFC 6480) prefix origin data and router keys from a trusted cache. This document describes a protocol to deliver validated prefix origin data and router keys to routers.

- Geoff Huston, Samuel Weiler, George Michaelson, Stephen Kent, “Resource Certificate PKI (RPKI) Trust Anchor Locator", draft-ietf-sidr-rfc6490-bis-01.txt, 2014-09-18, <http://tools.ietf.org/html/draft-ietf-sidr-rfc6490-bis-01.txt>

  This document defines a Trust Anchor Locator (TAL) for the Resource Certificate Public Key Infrastructure (RPKI).

- Pradosh Mohapatra, Keyur Patel, John Scudder, David Ward, Randy Bush, “BGP Prefix Origin Validation State Extended Community", draft-ietf-sidr-origin-validation-signaling-04.txt, 2014-02-14, <http://tools.ietf.org/html/draft-ietf-sidr-origin-validation-signaling-04.txt>

  As part of the origination AS validation process, it can be desirable to automatically consider the validation state of routes in the BGP decision process. The purpose of this document is to provide a specification for doing so. The document also defines a new BGP opaque extended community to carry the validation state inside an autonomous system to influence the decision process of the IBGP speakers.

- Samuel Weiler, Anuja Sonalker, Rob Austein, “A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", draft-ietf-sidr-publication-05.txt, 2014-02-12, <http://tools.ietf.org/html/draft-ietf-sidr-publication-05.txt>

This document defines a protocol for publishing Resource Public Key Infrastructure (RPKI) objects. Even though the RPKI will have many participants issuing certificates and creating other objects, it is operationally useful to consolidate the publication of those objects. This document provides the protocol for doing so.

- Randy Bush, “BGPsec Operational Considerations", draft-ietf-sidr-bgpsec-ops-05.txt, 2012-05-24, <http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-ops-05.txt>

  Deployment of the BGPsec architecture and protocols has many operational considerations. This document attempts to collect and present them. It is expected to evolve as BGPsec is formalized and initially deployed.

- Stephen Kent, Matthew Lepinski, Mark C. Reynolds, “Local Trust Anchor Management for the Resource Public Key Infrastructure", draft-ietf-sidr-ltamgmt-08.txt, 2013-04-05, <http://tools.ietf.org/html/draft-ietf-sidr-ltamgmt-08.txt>

  This document describes a facility to enable a relying party (RP) to manage trust anchors (TAs) in the context of the Resource Public Key Infrastructure (RPKI). It is common in RP software (not just in the RPKI) to allow an RP to import TA material in the form of self-signed certificates. However, this approach to incorporating TAs is potentially dangerous. (These self-signed certificates rarely incorporate any extensions that impose constraints on the scope of the imported public keys, and the RP is not able to impose such constraints.) The facility described in this document allows an RP to impose constraints on such TAs. Because this mechanism is designed to operate in the RPKI context, the most important constraints are the Internet Number Resources (INRs) expressed via RFC 3779 extensions. These extentions bind address spaces and/or autonomous system (AS) numbers to entities. The primary motivation for the facility described in this document is to enable an RP to ensure that INR information that it has acquired via some trusted channel is not overridden by the information acquired from the RPKI repository system or by the putative TAs that the RP imports. Specifically, the mechanism allows an RP to specify a set of overriding bindings between public key identifiers and INR data. These bindings take precedence over any conflicting bindings acquired by the putative TAs and the certificates downloaded from the RPKI repository system. This mechanism is designed for local use by an RP, but any entity that is accorded administrative control over a set of RPs may use this mechanism to convey its view of the RPKI to RPs within its jurisdiction. The means by which this latter use case is effected is outside the scope of this document.

The following internet-drafts, co-authored by members of the PARSONS team, have not been accepted by the SIDR working groups as work items.

- Edric Barnes, “Resource Public Key Infrastructure (RPKI) Resource Transfer Protocol and Transfer Authorization Object (TAO)”, draft-barnes-sidr-tao-00.txt, 2014-02-13, <http://tools.ietf.org/html/draft-barnes-sidr-tao-00.txt>

This document defines an extension to the rpki-updown protocol to provide support for transferring Internet Number Resources from one INR holder to another. Such transfers take place external to the RPKI, using procedures defined within and between RIRs. This protocol facilitates automation of the maintenance of RPKI data in the context of INR transfers. The protocol supports asynchronous transfers of live or unused INRs within an RIR or between RIRs. The scope of this protocol is limited to the transfer of Internet Number Resources within the Resource Public Key Infrastructure. In support of this protocol, this document also defines a new signed object type for the RPKI repository system, the Transfer Authorization Object (TAO).

- Randy Bush, "Responsible Grandparenting in the RPKI", draft-ymbk-rpki-grandparenting-04.txt, 2014-02-04, <http://tools.ietf.org/html/draft-ymbk-rpki-grandparenting-04.txt>

There are circumstances in RPKI operations where a resource holder's parent may not be able to, or may not choose to, facilitate full and proper registration of the holder's data. As in real life, the holder may form a relationship with their grandparent who is willing to aid the grandchild. This document describes simple procedures for doing so.

- Stephen Kent, David Mandelberg, "Suspenders: A Fail-safe Mechanism for the RPKI", draft-kent-sidr-suspenders-02.txt, 2014-07-03, <http://tools.ietf.org/html/draft-kent-sidr-suspenders-02.txt>

The Resource Public Key Infrastructure (RPKI) is an authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. The certification authorities (CAs) in the RPKI issue certificates to match their allocation of INRs. These entities are trusted to issue certificates that accurately reflect the allocation state of resources as per their databases. However, there is some risk that a CA will make inappropriate changes to the RPKI, either accidentally or deliberately (e.g., as a result of some form of "government mandate"). The mechanisms described below, and referred to as "Suspenders" are intended to address this risk.

Suspenders enables an INR holder to publish information about changes to objects it signs and publishes in the RPKI repository system. This information is made available via a file that is external to the RPKI repository, so that Relying Parties (RPs) can detect erroneous or malicious changes related to these objects. RPs can then decide, individually, whether to accept changes that are not corroborated by independent assertions by INR holders, or to revert to previously verified RPKI data.

- David Mandelberg, "Simplified Local internet nUmber Resource Management with the RPKI", draft-dseomn-sidr-slurm-01.txt, 2014-07-03, <http://tools.ietf.org/html/draft-dseomn-sidr-slurm-01.txt>

The Resource Public Key Infrastructure (RPKI) is a global authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. Internet Service Providers (ISPs) can use the RPKI to validate BGP route origination assertions. Some ISPs locally use BGP with private address space or private AS numbers (see RFC6890). These local BGP routes cannot be verified by the global RPKI, and SHOULD be considered invalid based on the global RPKI (see RFC6491). The mechanisms described below provide ISPs with a way to make local assertions about private (reserved) INRs while using the RPKI's assertions about all other INRs.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

API Application Programming Interface.

AS Autonomous System
AS is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet.

ASN Autonomous System Number. ASN is the number associated to a AS.

ASN.1 Abstract Syntax Notation One
ASN.1 is a language for describing structured information.

BGP Border Gateway Protocol
This is the routing protocol used to transfer reachability and routing information between Autonomous Systems (AS's) on the internet.

BGPSEC BGP Security
Additions to the BGP protocol to increase the security of the exchanged routing information.

BIRD The BIRD Internet Router Daemon is an open source routing software package

BPKI Business Public Key Infrastructure.

CA Certification Authority (x509)
An entity that issues digital certificates. The certificates are used to certify that the subject of the certificate owns the public key associated with the certificate. In the X.509 Public Key Infrastructure model, the CA is a trusted third party. That is, the owner of the certificate and the user, or relying party, of the certificate both trust the CA.

CMS Cryptographic Message Syntax
CMS is the IETF's standard for cryptographically protected messages.

ECDSA Elliptic Curve Digital Signature Algorithm (ECDSA)
is a variant of the Digital Signature Algorithm (DSA) which operates on elliptic curve groups. The EC variant provides smaller key sizes for the same security level. This algorithm is used for signing and authenticating within the BGPSEC path attribute in BGP UPDATE messages.

GUI Graphical User Interface.

IANA Internet Assigned Numbers Authority
The IANA manages the DNS Root Zone, coordinates allocations from the global IP and AS number spaces, and serves as the central repository for protocol name and number registries used in many Internet protocols.

IETF Internet Engineering Task Force [22]
The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet.

IRBE Internet Registry Back End.

IRDB Internet Registry Data Base.

IRR Internet Routing Registry
The union of world-wide routing policy databases that use the Routing Policy Specification Language.

ISP Internet Service Provider.

ODBC Open Database Connectivity
ODBC is a standard programming language middleware API for accessing database management systems.

ORM Object-Relational Mapper
ORM is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages. In Django, the data models are defined as Python classes.

RIR Regional Internet Registries
Regional Internet Registries (RIRs) manage, distribute, and register public Internet Number Resources within their respective regions.

ROA Route Origin Authorization
A ROA is a digitally signed object that provides a means of verifying that an IP address block holder has authorized an Autonomous System (AS) to originate routes to one or more prefixes within the address block.

RP Relying Party
An entity or individual that acts in reliance on certificates issued by a certification authority (CA).

RPA Relying Party Agreement
An agreement between a CA and its Relying Parties.

RPKI Resource Public Key Infrastructure
A Public Key Infrastructure (PKI) used to support attestations about Internet Number Resource (INR) holdings.

RPSL Routing Policy Specification Language
The language used to express routing policies in an Internet Routing Registry.

RPSTIR Relying Party Security Technology for Internet Routing
Pronounced ”rip-stir”. Using the global Resource Public Key Infrastructure (RPKI), RPSTIR securely generates a list of authorized prefix-origin AS pairs. This list can be used by the RPKI-RTR protocol, enabling routers to detect false origin announcements due to errors by network operators.

RTR RPKI to Router Protocol
The RPKI to router protocol, used to communicate validated prefix origin data from a trusted RPKI cache to a router. 13, 44

SIDR Secure Inter-Domain Routing (Working Group)
IETF Working Group that worked on creating BGPSEC [49].

SQL Structured Query Language
SQL is a query language used for managing data held in a relational database system.

SSH Secure Shell
SSH is a protocol for secure remote login and other secure network services over an insecure networks.

SSL Secure Sockets Layer
SSL is a protocol that provides communication security over the Internet.

SVN Apache Subversion (often abbreviated SVN, after the command name svn)
is a software versioning and revision control system distributed as free software under the Apache License. Developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation.

XML Extensible Markup Language
XML is a markup language defined by the W3C[53] that defines a set of rules for encoding documents in a format that is human and machine readable.

YaML YAML Ain't Markup Language
YaML is a human-readable data serialization format.

GLOSSARY OF TERMINOLOGY

Apache Apache is a commercial grade web server package.

Django Django is a free and open source web application framework written in Python, that encourages rapid development and clean, pragmatic design.

ghostbuster An RPKI signed object which contains contact information for a person responsible for the RPKI repository in which the object appears.

irdbd A sample implementation of an IR database daemon.

left-right The left-right protocol is two separate client/server protocols over separate channels between the RPKI engine and the IR back end (IRBE). The IRBE is the client for one of the subprotocols, the RPKI engine is the client for the other.

MySQL a widely used and popular open source database package.

OpenSSL a widely used, open source implementation of many cryptographic features, including support for X.509 Public Key Infrastructure, XML, and many different cryptographic algorithms.

prefix a contiguous block of Internet addresses, called a prefix because all the addresses share the same initial bit pattern.

pubd The publication engine daemon.

rcynic The primary validation tool in the rpki.net package.

Relying Party The entity which retrieves RPKI objects from repositories, validates them, and uses the result of that validation process as input to other processes, such as BGP security.

rootd A separate daemon for handling the root of an RPKI certificate tree.

RouteViews Route Views is a project founded by Advanced Network Technology Center at the University of Oregon to allow Internet users to view global BGP routing information from the perspective of other locations around the internet. Originally created to help Internet Service Providers determine how their network prefixes were viewed by others in order to debug and optimise access to their network, Route Views is now used for a range of other purposes such as academic research.

rpki-rtr A protocol to deliver validated prefix origin data to routers. Described in RFC 6810 [10].

rpki.net rpki.net is a project and website. The project provides a free, BSD License, open source, complete system for the Internet Registry or ISP. It includes separate components which may be combined to suit your needs.

rpkic A command line interface to control rpkid and pubd. 10, 44/ rpkid The main RPKI certificate issuance daemon.

rsync A file synchronization and file transfer program for Unix-like systems that minimizes network data transfer by using a form of delta encoding called the rsync algorithm.

rtr-origin rtr-origin is an implementation of the rpki-rtr protocol, including the rpki-rtr server, a test client and a utility for examining the content of the database rtr-origin generates.

RTRlib RPKI RTR Client C Library/ The RTRlib is an open-source C implementation of the RPKI/Router Protocol client [38].

subversion Apache Subversion (often abbreviated SVN, after the command name svn) is a software versioning and revision control system distributed as free software under the Apache License. Developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation.

up-down A RPKI certificate provisioning protocol which is expressed as a simple request/response interaction, where the client passes a request to the server, and the server generates a corresponding response. Described in RFC 6492[18].

X.509 an ITU-T standard for public key infrastructure (PKI) certificates and certificate revocation lists.